Carl von Ossietzky Universität Oldenburg
Fachbereich Informatik
Abteilung Informationssysteme
Prof. Dr. H.-J. Appelrath



DIPLOMARBEIT

Vorgehensmodell und Entwicklungsmethodik für virtuelle Labore

vorgelegt von:

Ansgar Scherp

Erstprüfer:

Prof. Dr. Hans-Jürgen Appelrath

Zweitprüfer:

Dipl.-Inform. Dietrich Boles

Oldenburg, den 26. August 2001

(Adobe PDF Version)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Im Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme (OFFIS) wird in der Arbeitsgruppe Multimedia-Systeme von Prof. Dr. H.-J. Appelrath seit September 1997 ein multimediales gentechnisches Praktikum (*GenLab*) entwickelt, mit dem gentechnische Experimente in einer virtuellen Umgebung auf einem handelsüblichen Computer-System durchgeführt werden können (siehe GenLab - Internetseiten, 2001). Dazu wurden die Bestandteile eines realen Genlabors auf ein *virtuelles Labor* in einem Computer-System abgebildet. *GenLab* erlaubt eine interaktive und simulationsbasierte Durchführung der Experimente am Bildschirm. Daneben veranschaulicht eine umfangreiche Wissenskomponente die entsprechenden molekularbiologischen Abläufe und erläutert den richtigen Umgang mit den Laborgeräten und Substanzen. Der Grundgedanke von *GenLab* ist es, Experimente im virtuellen Labor so realitätsgetreu wie möglich durchführen zu können. Dabei kommen die Vorteile gegenüber realen Laborpraktika in Form einer ständigen Fehlerüberwachung, Steuerungs- und Korrekturmöglichkeiten sowie die Verknüpfung zu einer Wissenskomponente zum Tragen (Appelrath und Schlattmann, 1999, Seite 92 bis 95). Allerdings ist auch die Erstellung eines virtuellen Labors wie *GenLab* sehr zeitaufwendig und teuer. Aus diesem Grund schließt sich an *GenLab* das Projekt *VirtLab* an, in dessen Rahmen von der konkreten Implementierung des virtuellen Labors *GenLab* abstrahiert wird. In *VirtLab* werden Methoden und Werkzeuge, also Vorgehensmodelle, Notationen, Entwurfsmuster, Frameworks und Spezifikationssprachen, zur Entwicklung allgemeiner multimedialer virtueller naturwissenschaftlich-technischer Labore und Praktika erarbeitet. Erklärtes Ziel von *VirtLab* ist es, die schnelle und kostengünstige Produktion qualitativ hochwertiger virtueller Labore zu ermöglichen.

1.1. Motivation und Einordnung

Professionelle Software-Entwicklung ist heute ohne den Einsatz eines Vorgehensmodells nicht mehr vorstellbar. Erstaunlicherweise finden allerdings die aufgrund der Software-Krise in den sechziger Jahren entstandenen Vorgehensmodelle und Entwicklungsmethoden der Software-Technik in der Entwicklung von Multimedia-Systemen kaum Verwendung (siehe Boles u. a., 1998). So wird denn auch die Vorgehensweise bei der Entwicklung von Multimedia-Systemen von den Unternehmen selbst als eher *ad hoc* bezeichnet. Besonders der Bereich der multimedialen Lehr- und Lernsysteme stellt jedoch sehr hohe Anforderungen an die Projektorganisation, Planung und Konzeption. Ein Grund hierfür ist die Heterogenität

des Entwicklerteams, das typischerweise aus Fachexperten der zu modellierenden Domäne, (Fach-)Didaktikern, Medienspezialisten und Informatikern besteht. Daher verwundert es auch nicht, dass in der Umfrage von (Nagl u. a., 1999) im Bezug auf die Defizite und Wünsche zur Entwicklung von multimedialen Lehr- und Lernsystemen unter anderem verbesserte Prozessmodelle genannt werden. In *GenLab* wurde des Weiteren festgestellt, dass neben einem Vorgehensmodell insbesondere eine geeignete Entwicklungsmethodik für den Anwendungsfall virtuelle Labore fehlt. So sind die bestehenden Vorgehensmodelle zur Entwicklung von multimedialen Lehr- und Lernsystemen zu abstrakt, als dass sie für die Entwicklung von virtuellen Laboren nutzbringend eingesetzt werden könnten. Der hohe Grad der Interaktionsmöglichkeiten in virtuellen Laboren sowie die Erstellung und Verarbeitung wichtiger Artefakte, wie zum Beispiel die Versuchsprotokolle und das Drehbuch, werden von diesen Vorgehensmodellen nicht oder nur zum Teil berücksichtigt.

1.2. Anforderungen

Im Rahmen dieser Arbeit wird daher ein Vorgehensmodell und eine Entwicklungsmethodik für den Anwendungsfall virtuelle Labore erstellt. Das Vorgehensmodell beschreibt, wann bei der Entwicklung virtueller Labore welche Entwickler welche Aktivitäten durchzuführen haben. Die zugehörige Entwicklungsmethodik legt fest, wie diese Aktivitäten durchzuführen sind und welche Artefakte dabei erstellt oder verwendet werden. Dabei ist der hohe Grad an Interdisziplinarität des Entwicklerteams und den damit verbundenen sehr hohen Kommunikationsbedarf zwischen den Projektmitarbeitern zu berücksichtigen. Des Weiteren ist der Software-Entwicklungsprozess in mehrere Phasen mit festdefinierten Meilensteinen einzuteilen. Es sind die zur Entwicklung von virtuellen Laboren durchzuführenden Aktivitäten und die zu erstellenden Artefakte zu identifizieren. Dazu sind Anleitungen, Richtlinien und Methoden zur Durchführung der einzelnen Aktivitäten und zur Erstellung der jeweiligen Artefakte bereit zu stellen. Wichtigste Anforderung ist jedoch die Gewährleistung der Praxistauglichkeit. Der Abstraktionsgrad des Vorgehensmodells muss gerade so hoch sein, dass es für die Entwicklung beliebiger virtueller Labore anwendbar ist. Es darf jedoch nicht den hohen Abstraktionsgrad der Vorgehensmodelle erreichen, die für die Entwicklung von beliebigen multimedialen Lehr- und Lernsystemen einsetzbar sind.

1.3. Aufbau der Arbeit

Der weitere Aufbau der Arbeit gliedert sich wie folgt: In Kapitel 2 werden zunächst einige Begriffe geklärt, die im Kontext dieser Arbeit wichtig sind.

Danach wird in Kapitel 3 eine umfangreiche Analyse der Problematik eines geeigneten Vorgehensmodells für virtuelle Labore durchgeführt. Dazu werden bereits bekannte und erprobte Vorgehensmodelle der Software-Technik analysiert und diese in Hinblick auf Anwendbarkeit für virtuelle Labore bewertet. Anschließend wird ein Vorgehensmodell als Lösungsansatz für den Anwendungsfall virtuelle Labore herausgearbeitet.

In Kapitel 4 erfolgt zu diesem Lösungsansatz zunächst ein kurzer Überblick und eine Beschreibung der Phasen und Meilensteine. Anschließend werden in Kapitel 5 die am Software-Entwicklungsprozess beteiligten Rollen beschrieben. Danach wird in Kapitel 6 das Metaobjektmodell für virtuelle Labore als Grundbaustein der Entwicklungsmethodik eingeführt. In Kapitel 7 werden dann die Workflows des Vorgehensmodells zusammen mit den zugehörenden Aktivitäten und Artefakten der Entwicklungsmethodik für virtuelle Labore beschrieben.

Im abschließenden Kapitel 8 wird eine Bewertung der Arbeit vorgenommen und es erfolgt ein Ausblick bezüglich zukünftiger Arbeiten. Außerdem befindet sich im Anhang zu dieser Arbeit ein Beispiel eines Anwendungsfalls (siehe Anhang A), Checklisten zu den wichtigsten Artefakten der Entwicklungsmethodik (siehe Anhang B), ein Beispiel einer Prozesssicht auf das Vorgehensmodell (siehe Anhang C), ein ausführliches Glossar (siehe Anhang D), ein Literaturverzeichnis und ein umfassender Index.

2. Begriffsdefinitionen

Ziel dieses Kapitels ist es, die Bedeutung einiger Begriffe, die im Rahmen dieser Arbeit wichtig sind, zu klären. Im Einzelnen sind das die Begriffe

- *Vorgehensmodell*,
- *Entwicklungsmethodik*,
- *Multimedia-System*,
- *Lehr- und Lernsystem*,
- *multimediales Lehr- und Lernsystem*,
- *Simulator*,
- *intelligentes Tutorsystem* und
- *virtuelles Labor*.

Die Begriffe Vorgehensmodell und Entwicklungsmethodik werden in den Kapiteln 2.1 und 2.2 aufgeführt, da diese grundlegend für die vorliegende Arbeit sind. In den Kapiteln 2.3 und 2.4 werden die Begriffe Multimedia-System und Lehr- und Lernsystem beschrieben, da diese ebenfalls grundlegend sind und nicht als allgemein bekannt vorausgesetzt werden können. Des Weiteren basiert auf diesen beiden Begriffen der Begriff des multimedialen Lehr- und Lernsystems, der in Kapitel 2.5 festgelegt wird. Die Begriffe Simulator und intelligentes Tutorsystem werden in den Kapiteln 2.6 und 2.7 aufgeführt, da diese die beiden wesentlichen Bestandteile eines virtuellen Labors darstellen. Abschließend wird in Kapitel 2.8 der Begriff des virtuellen Labors als ein Spezialfall multimedialer Lehr- und Lernsysteme definiert.

2.1. Vorgehensmodell

Nach (Balzert, 2000b, Seite 71) wird unter einem Vorgehensmodell beziehungsweise Prozessmodell ein allgemeiner Entwicklungsplan verstanden, der das generelle Vorgehen beim Entwickeln eines Software-Produkts festlegt. Neben dieser sehr allgemeinen Beschreibung des Begriffs des Vorgehensmodells existieren in der Literatur auch konkretere Definitionen.

So beschreibt ein Vorgehensmodell nach (oose.de GmbH, 1999) auf abstrakte oder generische Weise, wie ein System entwickelt wird, das heißt in welcher Reihenfolge welchen Aktivitäten und Ergebnissen nachgegangen wird. Zusammen mit einer Notation und Sprache wie der *Unified Modeling Language*[1] (UML) sowie einer Reihe von Managementpraktiken entsteht daraus eine Entwicklungsmethodik (kurz: *Methodik*). Demnach beschreibt ein Vorgehensmodell also nicht nur das Vorgehen bei der Software-Entwicklung, sondern wird in Verbindung mit einer Notation und bestimmten Managementpraktiken zu einer Entwicklungsmethodik. Eine klare Trennung zwischen Vorgehensmodell und Entwicklungsmethodik ist in dieser Definition nicht mehr erkennbar. Aus diesem Grund wird im Rahmen dieser Arbeit unter einem Vorgehensmodell die aus (Appelrath u. a., 1998, Seite 108) entnommene Definition verstanden. Demnach beschreibt ein Vorgehensmodell den Lebenszyklus eines Software-Produkts in Form von Aktivitäten. Durch das Vorgehensmodell wird festgelegt, in welcher Reihenfolge welche Aktivitäten durchgeführt werden können und welche zeitlichen Überschneidungen zulässig sind. Ein Vorgehensmodell liefert außerdem Informationen über die für eine Aktivität relevanten Objekte und die einzusetzenden *Entwicklungsmethoden und -werkzeuge*.

2.2. Entwicklungsmethodik

Allgemein wird nach (Langenscheidts Fremdwörterbuch, 2001) unter Methodik die Lehre vom planmäßigen Unterrichten beziehungsweise die Lehre von den wissenschaftlichen Methoden und ihrer Anwendung verstanden (vergleiche auch Brockhaus-Enzyklopädie, 1992). Im Bezug auf die Entwicklung von Software-Systemen bezeichnet die Entwicklungsmethodik eine Verfahrenslehre zum systematischen Entwickeln eines Software-Systems unter Berücksichtigung aller Einflussfelder und Lebensphasen von der Vision bis zum Produkttod (vergleiche Groos und Muthmann, 1995, Seite 192). Daran angelehnt wird im Rahmen dieser Arbeit unter der Entwicklungsmethodik konkret die Menge der Vorschriften zur Durchführung von Aktivitäten und Repräsentation entsprechender Ergebnisse verstanden.

2.3. Multimedia-System

Unter einem Multimedia-System (MMS) wird ein Software-System verstanden, das mindestens gekennzeichnet ist durch die rechnergesteuerte, integrierte Darstellung und Kommunikation sowie optional die Erzeugung, Manipulation und Speicherung von unabhängigen Informationen, die in mindestens einem kontinuierlichen und einem diskreten Medium kodiert sind (vergleiche dazu Steinmetz, 1999, Seite 13). Beispiele für kontinuierliche Medien sind Audio- und Videosequenzen sowie Animationen. Diskrete Medien sind beispielsweise Texte, Graphiken und Diagramme.

[1] Siehe (Object Management Group, 1997–2001)

2.4. Lehr- und Lernsystem

Ein Lehr- und Lernsystem (LLS) im klassischen Sinne ist ein Software-System, das eigenständiges Lernen ohne unmittelbare Einwirkung eines Lehrers gestattet. Lehr- und Lernsysteme bieten Informationen, verlangen vom Lerner[2] eigene Aktivität durch Beantwortung von Kontrollfragen, auf die eine unmittelbare Rückmeldung gegeben wird (vergleiche die behavioristischen und kybernetischen Ansätze der Lerntheorien in Kerres, 1998, Seite 45 bis 56). Vor allem werden in Lehr- und Lernsystemen durch Programmverzweigungen zusätzliche Lösungshilfen angeboten. Dies geschieht in Form neuer Aufgaben und meist in kleineren Schritten. Das Grundprinzip eines Lehr- und Lernsystems ist die Gliederung des Stoffes in eine Sequenz von Instruktions- und Testeinheiten (vergleiche Brockhaus-Enzyklopädie, 1992, Stichwort: Lehr- und Lernmaschinen und Lehrautomaten). Lehr- und Lernsysteme entlasten den Lehrer bei der Lehre insofern, dass auch ohne die Anwesenheit eines Lehrers mit Hilfe eines Lehr- und Lernsystems der Wissensstand des Lerners geprüft und bewertet werden kann. Die Weiterentwicklung der Lehr- und Lernsysteme zielt auf tutorielle Systeme (sogenannte intelligente Tutorsysteme, siehe Kapitel 2.7) ab, die dem Lerner selbst Hinweise geben, wie er vom System am besten unterrichtet werden kann (siehe auch Blumstengel, 1998, Kapitel 1.2).

2.5. Multimediales Lehr- und Lernsystem

Ein multimediales Lehr- und Lernsystem (MMLLS) ist nach (Nagl u. a., 1999) ein multimediales System, dessen Zweck darin besteht, den Lerner beim Lernen und den Lehrer bei der Lehre zu unterstützen. Multimediale Lehr- und Lernsysteme variieren in technischer wie inhaltlicher Hinsicht erheblich. Aus technischer Sicht handelt es sich um ein Multimedia-System wie in Kapitel 2.3, das zum Beispiel eine multimediale CD-ROM oder multimediale Dokumente auf einem WWW-Server sein kann. Dabei können sowohl die beteiligten Medien und Formate als auch deren Mischung variieren. Aus inhaltlicher Sicht entspricht ein multimediales Lehr- und Lernsystem einem Lehr- und Lernsystem nach Kapitel 2.4 und kann einen kompletten Kurs, eine einzelne Vorlesung oder Schulstunde, aber auch nur ein Fragment davon abdecken. Ein multimediales Lehr- und Lernsystem kann in der schulischen Ausbildung, an Hoch- und Fachschulen, in der beruflichen Erst- und Zweitausbildung sowie in der beruflichen Fortbildung eingesetzt werden (siehe auch BmBF, 2001).

2.6. Simulator

Nach der (Brockhaus-Enzyklopädie, 1992) wird in der Informatik unter einem Simulator ein Software-System verstanden, das einen Prozess oder das Verhalten eines Systems auf einem Computer-System darstellt oder nachbildet (*simuliert*). Eine *Simulation* ist demnach eine

[2] Siehe Glossar

modellhafte Darstellung oder Nachbildung bestimmter Aspekte eines vorhandenen oder zu entwickelnden Systems auf einem Computer-System mit Hilfe mathematischer oder abstrakter Modelle (*Simulationsmodelle*).[3] Dabei wird insbesondere das Zeitverhalten des Systems betrachtet, das heißt wie sich der Zustand des Systems über die Zeit ändert (vergleiche Duden »Informatik«, 1993, Seite 648).

Die Simulation erlaubt die Untersuchung oder Manipulation von Abläufen, die man in der Wirklichkeit zum Beispiel aus Zeit-, Kosten- und Gefahrengründen nicht durchführen kann. Im Simulationsmodell spiegeln sich die wesentlichen Eigenschaften der zu simulierenden Vorgänge und ihre gegenseitige Beeinflussung wider. Mit Hilfe einer Simulation kann eine Prognose für die Wettervorhersage generiert, ein Pilot in einem Flugsimulator ausgebildet oder ein Experiment in einem virtuellen Labor durchgeführt werden. Alle Ergebnisse einer Simulation beziehen sich nur auf das Simulationsmodell. Inwieweit die Ergebnisse auf die Wirklichkeit übertragen werden können, hängt daher entscheidend davon ab, wie gut die Wirklichkeit durch das Modell nachgebildet wird (vergleiche Groos und Muthmann, 1995, Seite 362 f. und Seite 503). Ob das zugrunde liegende Modell als zufriedenstellend angesehen wird, hängt vom Zweck und der gewünschten Genauigkeit einer Simulation ab (aus Duden »Informatik«, 1993).

Simulationen lassen sich einteilen in

- *deterministische Simulation* und
- *stochastische Simulation*,

sowie in

- *zeitkontinuierliche Simulation* und
- *zeitdiskrete Simulation*.[4]

Bei der deterministischen Simulation sind alle am Modell beteiligten Größen, die das Verhalten eines Modells bestimmen, exakt definiert oder aufgrund mathematischer Zusammenhänge berechenbar. In stochastischen Simulationen werden in den abstrakten Modellen auch zufallsbedingte Größen verwendet. Für die Erzeugung der Zufallsgrößen kommen häufig Zufallszahlengeneratoren zum Einsatz (aus Brockhaus-Enzyklopädie, 1992; Duden »Informatik«, 1993). Werden fortlaufend sämtliche Werte der Simulation neu berechnet, so wird von einer zeitkontinuierlichen Simulation gesprochen. Dagegen werden in zeitdiskreten Simulationen nur ausgewählte Werte zu bestimmten, klar unterscheidbaren Zeitpunkten neu berechnet (siehe Brockhaus-Enzyklopädie, 1992).

---

[3] Anmerkung: In (Brockhaus-Enzyklopädie, 1992) wird der Begriff der Simulation zunächst allgemeiner gefasst. Dort wird prinzipiell zwischen zwei Extremen von Simulationen unterschieden, nämlich der Simulation mittels real existierenden physikalischen oder technischen Modellen und der hier betrachteten Simulation mittels mathematischer oder abstrakter Modelle. Dazwischen gibt es viele Grade der Mischung beider Prinzipien. Die Variante der real existierenden Simulation, wie zum Beispiel der *Crash-Test* für Kraftfahrzeuge, ist hier nicht weiter von Interesse und wird daher nicht weiter betrachtet.

[4] Siehe auch (Steinhausen, 1994, Seite 6 f.)

Zur Programmierung von Simulationen existieren spezielle Programmiersprachen, wie zum Beispiel *SIMULA* oder *General Purpose Simulation System*, in denen bestimmte Simulationskonzepte einfach formuliert werden können (Duden »Informatik«, 1993).

2.7. Intelligentes Tutorsystem

Intelligente Tutorsysteme[5] (ITS) stellen eine Weiterentwicklung der in Kapitel 2.4 beschriebenen klassischen Lehr- und Lernsysteme dar. Nach (Groos und Muthmann, 1995, Seite 565) unterscheiden sich intelligente Tutorsysteme von klassischen Lehr- und Lernsystemen durch die unterstützte Lernstufe, das heißt das Niveau der vermittelten Kenntnisse und Fähigkeiten. Während konventionelle Lehr- und Lernsysteme vor allem Reproduktionswissen vermitteln, zielen intelligente Tutorsysteme darauf ab, dem Lerner selbst Hinweise zu geben, wie er vom System am besten unterrichtet werden kann (*adaptive Hilfe*). Der Lerner soll mit Hilfe eines Tutorsystems die Fähigkeit erwerben, Probleme aus dem Lehrbereich eigenständig und schrittweise zu lösen (*Hypothesentesten*). Für den hier betrachteten Anwendungsfall virtuelle Labore bedeutet dies zum Beispiel den Ablaufplan des durchgeführten Experiments zu erlernen und zu verstehen (vergleiche die kognitivistischen und situierten Ansätze der Lerntheorien in Kerres, 1998, Seite 56 bis 75).

Ein intelligentes Tutorsystem (vergleiche Schulmeister, 1997; Groos und Muthmann, 1995; Blumstengel, 1998) hat explizite Kenntnis über den aktuellen Wissensstand des Lerners (*Lernermodul*), die zu vermittelnden Lerninhalte aus einem Wissensgebiet (*Expertenmodul*), ein didaktisches Konzept zur Vermittlung der Lerninhalte (*Lehrermodul*) sowie eine Komponente zur Darstellung der Lerninhalte (*Benutzungsoberfläche*).

Abbildung 2.1 zeigt schematisch die zentralen Bestandteile der allgemeinen ITS-Architektur. Das Lernermodul beobachtet das Problemlösungsverhalten des Lerners. Außerdem protokolliert es den Stand und die Veränderung der Merkmale im Verlauf des Lösungsprozesses. Das Ergebnis des Lernermoduls ist die computergerechte Darstellung des Lerners in Form eines *Lernermodells*. Im Expertenmodul ist das Wissen von Experten zum Beispiel mit Hilfe eines Expertensystems modelliert. Daraus ergibt sich das *Expertenmodell*, in dem die Lerninhalte, die mit dem intelligenten Tutorsystem vermittelt werden sollen, enthalten sind. Das Lehrermodul entscheidet anhand eines *Lehrermodells*, wann welche Lerninhalte von der Benutzungsoberfläche des intelligenten Tutorsystems dem Lerner angeboten werden. Außerdem ist es dafür verantwortlich zu entscheiden, wann und wie oft das intelligente Tutorsystem den Lerner unterbricht. Für die eigentliche Darstellung der Lerninhalte und die Interaktion mit dem Lerner mittels Computer-Maus und Tastatur ist ebenfalls die Benutzungsoberfläche zuständig (siehe auch Kunz und Schott, 1987; Lusti, 1992; Liening, 1992; Witschital, 1990).

[5] Auch: *Intelligente Tutorielle Systeme* (siehe Schröder, 1996)

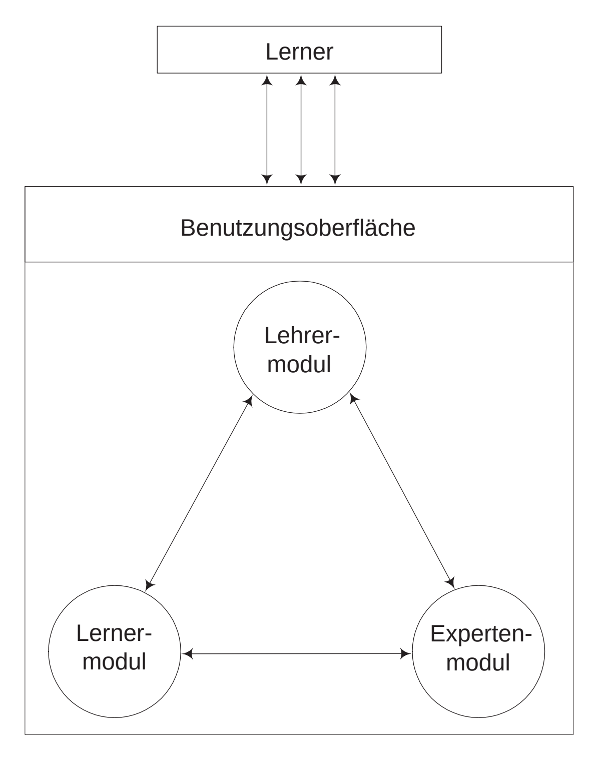


Abbildung 2.1.: Die allgemeine ITS-Architektur (nach Schröder, 1996)

2.8. Virtuelles Labor

Der Begriff des virtuellen Labors findet in der Literatur keine einheitliche Verwendung. So existieren mehrere voneinander abweichende und widersprüchliche Definitionen. Diese werden im Folgenden aufgeführt und es wird eine für diese Arbeit gültige Definition herausgearbeitet.

Oft werden bereits textuelle WWW-Seiten mit integrierten Animationen, *Java-Applets* oder *Shockwave-Filmen*, die zur Simulation von Geräten oder Prozessen dienen, als virtuelle Labore bezeichnet.[6]

In (Internet2, 1997–2000)[7] wird ein virtuelles Labor dagegen als eine heterogene, verteilte Problemlöseumgebung angesehen, die es einer über die Welt verteilten Gruppe von Wissenschaftlern ermöglicht, auf einer gemeinsamen Grundlage zusammenzuarbeiten. Wie in jedem anderen Labor sind auch hier die eingesetzten Werkzeuge und Methoden domänenabhängig. Die allgemeinen Anforderungen an die zugrunde liegende Infrastruktur, das heißt an die benötigten Kommunikations-Technologien, sind jedoch in allen Wissenschaftsfächern gleich.

[6] Zum Beispiel das *virtuelle Chemielabor* von Thomas Seilnacht (siehe: http://www.seilnacht.tuttlingen.com/Chemie.htm) oder das *Virtual Lab: Exploring Genetics* (siehe: http://library.thinkquest.org/11375/index.html).

[7] Anmerkung: Eine 1996 in den USA gegründete Initiative zur Weiterentwicklung des Internet, derzeit bestehend aus über 180 Universitäten sowie Partnerschaften in der Industrie und Regierung.

Im Rahmen von *VirtLab* werden Methoden und Werkzeuge zur Entwicklung allgemeiner multimedialer, virtueller, naturwissenschaftlich-technischer Labore und Praktika (kurz: virtueller Labore) erarbeitet. Dort wird unter einem virtuellen Labor die multimediale Nachbildung eines realen Labors im Rechner verstanden. Die Bestandteile eines virtuellen Labors entsprechen dem eines realen Labors und sind zum Beispiel virtuelle Laborgeräte, virtuelles Zubehör und virtuelle Substanzen. Ein virtuelles Labor kann aus verschiedenen Arbeitsflächen und Räumen bestehen. Die Nutzer können sich mit Hilfe von Maus und Tastatur im virtuellen Labor bewegen und dort Laborgeräte bedienen und Experimente durchführen. Besonderes Kennzeichen von virtuellen Laboren ist die hohe Interaktivität der Benutzungsoberfläche. Laborgeräte, Zubehör und Substanzen können in nahezu beliebiger Reihenfolge bedient und verwendet werden. Dabei können die virtuellen Laborgeräte und Substanzen zum Beispiel mit Hilfe einer Transportleiste zwischen den verschiedenen virtuellen Arbeitsflächen transportiert werden. Ein virtuelles Labor dient insbesondere zum Erlernen der teilweise sehr komplexen Arbeitsschritte und der benötigten Techniken naturwissenschaftlich-technischer Experimente. Dazu steht im virtuellen Labor zum Beispiel ein Anleitungsfenster zur Verfügung, in dem die einzelnen Arbeitsschritte eines Versuchsablaufs dargestellt werden (vergleiche Hasler und Schlattmann, 2001). Ein Beispiel einer Arbeitsfläche eines virtuellen Labors zeigt die aus *GenLab* entnommene Abbildung 2.2.

Zusammenfassend lässt sich also festhalten, dass ein virtuelles Labor im Sinne von *GenLab* beziehungsweise *VirtLab* ein multimediales, simulationsbasiertes, exploratives Lehr- und Lernsystem darstellt (vergleiche dazu auch das *Explorationen-Konzept* in Nowaczyk, 2000, Seite 29).[8] Darauf aufbauend ergibt sich die im Rahmen dieser Arbeit gültige Begriffsdefinition eines virtuellen Labors. Ein virtuelles Labor ist ein Software-System, das im Wesentlichen aus

- einem Simulator (siehe Kapitel 2.6) und
- einem intelligenten Tutorsystem (siehe Kapitel 2.7)

besteht. Dabei wird der Simulator nicht als reines Werkzeug verwendet, sondern ergibt in Verbindung mit dem Tutorsystem ein multimediales Lehr- und Lernsystem.

Denkbare konkrete Einsatzbereiche virtueller Labore sind vor allem die klassischen naturwissenschaftlichen Fachgebiete wie Biologie, Physik und Chemie — beginnend bei ihren Grundlagenthemen bis hin zu den Spezialgebieten[9] — und deren verschiedensten interdisziplinären Themenbereichen. Weitere mögliche Anwendungsfelder für virtuelle Labore sind zum Beispiel ein Pneumatiklabor in der Elektrotechnik oder ein Hardware-Labor

[8]Anmerkung: Die Ansätze von (Internet2, 1997–2000) und *VirtLab* werden ebenfalls von dem *Verbund Virtuelles Labor* (VVL, siehe: http://www.vvl.de/) verfolgt. Ausgehend von realen Hochschullaboratorien mit Robotern, Laborgeräten und Werkzeugmaschinen werden virtuelle Laboratorien und Laborexperimente entwickelt, die einerseits von Telenutzern über das Internet ferngesteuert werden können und andererseits die Möglichkeit bieten, den virtuellen Arbeitsprozess beziehungsweise das virtuelle Laborexperiment interaktiv zu steuern.

[9]Zum Beispiel im Bereich der Biologie die Gentechnik, umgesetzt im Projekt *GenLab* (siehe GenLab - Internetseiten, 2001)

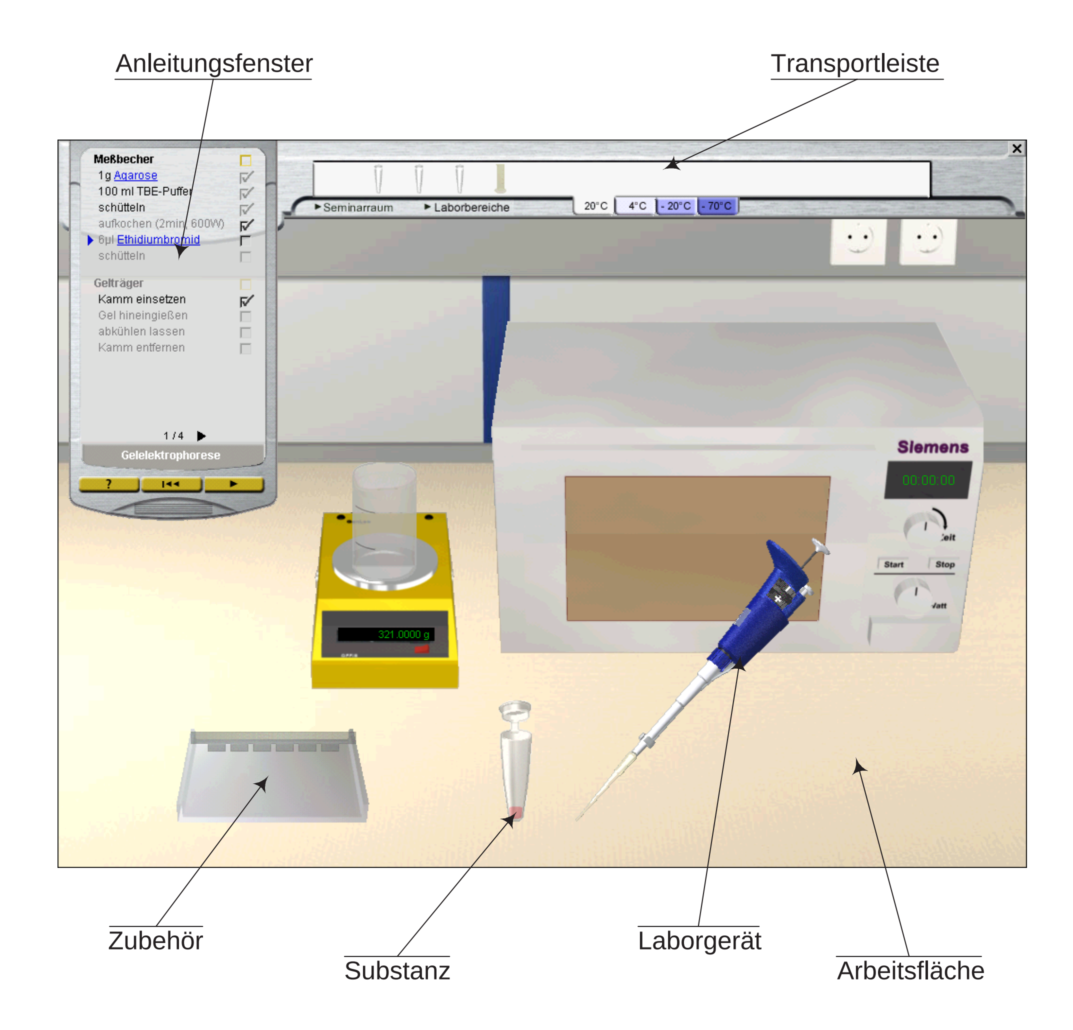


Abbildung 2.2.: Beispiel einer Arbeitsfläche eines virtuellen Labors (aus *GenLab*)

zur Entwicklung integrierter Schaltkreise in der Informatik. Eine ausführliche Analyse und Beschreibung weiterer Beispiele von virtuellen Laboren im Sinne der hier gültigen Definition sind zum Beispiel in (Elfreich, 1999, Seite 9 bis 29) sowie in (Kronberg, 1998; Jacob und Kronberg, 1999) und (Nowaczyk, 2000, Seite 29 bis 32) enthalten.

3. Problemanalyse

In diesem Kapitel wird eine umfangreiche Analyse der Problematik eines geeigneten Vorgehensmodells für virtuelle Labore durchgeführt. Aus der Literatur und Anwendung der Software-Technik sind bereits zahlreiche Vorgehensmodelle bekannt und hinreichend erprobt. Daher werden in Kapitel 3.1 zunächst einige Bewertungskriterien für Vorgehensmodelle in Hinblick auf den Anwendungsfall virtuelle Labore erarbeitet und ein entsprechendes Bewertungsschema aufgestellt. In Kapitel 3.2 werden einige Vorgehensmodelle der klassischen Software-Technik beschrieben. Danach folgen in Kapitel 3.3 weitere Vorgehensmodelle, die speziell auf die Entwicklung von Multimedia-Systemen, Lehr- und Lernsystemen und multimedialen Lehr- und Lernsystemen zugeschnitten sind. Darüber hinaus werden in Kapitel 3.4 einige Vorgehensmodelle aus der modernen Software-Technik vorgestellt. Am Ende der Kapitel 3.2, 3.3 und 3.4 werden die dort vorgestellten Vorgehensmodelle anhand des Bewertungsschemas bezüglich der Eignung für die Entwicklung virtueller Labore geprüft. Abschließend wird in Kapitel 3.5 eine Strategie zur Erstellung eines geeigneten Vorgehensmodells für virtuelle Labore erarbeitet.

3.1. Bewertungskriterien

Für die Bewertung der Vorgehensmodelle im Hinblick auf den Anwendungsfall virtuelle Labore sind eine Reihe von verschiedenen Aspekten zu berücksichtigen. Diese lassen sich unterteilen in *organisatorische*, *multimediale*, *didaktische* und *software-technische Aspekte*. Diese werden in den nachfolgenden Kapiteln 3.1.1 bis 3.1.4 beschrieben.

3.1.1. Organisatorische Aspekte

Die organisatorischen Aspekte betreffen die Aufteilung des Entwicklerteams während der Erstellung des virtuellen Labors, sowie die Einbindung des Auftraggebers und der Endanwender in den Software-Entwicklungsprozess. Daher lassen sich bezüglich der Organisationsstruktur folgende Bewertungskriterien identifizieren:

- **Heterogenität des Entwicklerteams** (kurz: Heterogenität des Teams): Virtuelle Labore bilden einen eigenen Bereich der multimedialen Lehr- und Lernsysteme, in dem bestimmte Aspekte der Entwicklung, wie zum Beispiel die Zusammenarbeit mit den

Fachexperten, noch stärker in den Vordergrund gerückt werden (siehe Aden u. a., 1999, Seite 34). So zeichnet sich die Entwicklung von virtuellen Laboren gerade wegen der zwangsläufigen Heterogenität des Entwicklerteams durch sehr hohe Anforderungen an die Projektorganisation aus. Das Entwicklerteam eines virtuellen Labors besteht typischerweise aus *Fachexperten*[1] *der Domäne des virtuellen Labors*, die im Folgenden abkürzend mit *Fachexperten* bezeichnet werden, *Fachdidaktikern*, *Medienspezialisten* und *Informatikern*. Die hohen Anforderungen an die Projektorganisation begründen sich darin, dass jede der vier Fachrichtungen eine eigene Fachsprache verwendet, die von den anderen Projektmitarbeitern und insbesondere dem Projektleiter verstanden werden muss (vergleiche Aden u. a., 1998, Seite 9). Es ist also zu bewerten, inwiefern die Heterogenität des Entwicklerteams von dem Vorgehensmodell unterstützt wird oder wie aufwendig die Integration eines geeigneten Rollenkonzeptes (siehe ANSSTAND e.V., 2001; Pasch, 1994) in das Vorgehensmodell ist.

- **Einbeziehung des Auftraggebers** (kurz: Einbez. des Auftraggebers): Ein weiteres wichtiges Bewertungskriterium ist die Einbeziehung des Auftraggebers in den Software-Entwicklungsprozess. Neben der Erstellung der Anforderungsdefinition sollte der Auftraggeber in regelmäßigen Abständen das virtuelle Labor in Form eines Pilotsystems evaluieren können. Von besonderem Interesse ist dabei für den Auftraggeber, ob die Vorgaben des *Corporate Designs* eingehalten werden, falls dieses in dem Unternehmen definiert ist (siehe Yass, 2000, Seite 17 und 88). Die Ergebnisse der Evaluation durch den Auftraggeber fließen anschließend direkt in die weitere Entwicklung des virtuellen Labors ein. Es ist also die Häufigkeit und der Zeitpunkt der Einbeziehung des Auftraggebers in den Software-Entwicklungsprozess zu betrachten.

- **Einbeziehung der Endanwender** (kurz: Einbez. der Endanwender): Gerade für die Akzeptanz und den Erfolg eines virtuellen Labors ist die Einbeziehung der zukünftigen Endanwender beziehungsweise Lerner in den Software-Entwicklungsprozess besonders wichtig. Die Lerner sollten in regelmäßigen Abständen das virtuelle Labor in Form eines Pilotsystems bezüglich Interaktionsmöglichkeiten, Gestaltung der Benutzungsoberfläche und Verständlichkeit der Lerninhalte evaluieren können. Dadurch kann eine realistische Bewertung vorgenommen werden, inwiefern das virtuelle Labor von den Lernern akzeptiert wird und wie gut die Lerner im virtuellen Labor lernen, das heißt die vermittelten Lerninhalte aufnehmen und nachvollziehen. Auch können so Unterschiede in der Durchführung eines Laborpraktikums an verschiedenen Universitäten oder Instituten ermittelt und im virtuellen Labor entsprechend berücksichtigt werden. Die Ergebnisse der Evaluation durch die Endanwender fließen dabei zurück in den weiteren Entwicklungsprozess des virtuellen Labors.

---

[1] Auch: Sachexperte (siehe Kerres, 1998)

3.1.2. Multimediale Aspekte

Da ein virtuelles Labor nach Kapitel 2.8 ein spezielles multimediales Lehr- und Lernsystem darstellt, sind von einem Vorgehensmodell für virtuelle Labore auch die multimedialen Aspekte zu berücksichtigen. Diesbezüglich lassen sich die beiden folgenden Bewertungskriterien identifizieren:

- **Erstellung der Benutzungsoberfläche** (kurz: Benutzungsoberfläche): Bei der Entwicklung eines virtuellen Labors wird eine sehr hohe Gewichtung auf die multimediale Gestaltung der Benutzungsoberfläche gelegt. Ziel ist eine möglichst realistische Darstellung der Laborkomponenten und Versuchsabläufe zu erreichen, damit die im virtuellen Labor erlernten Handlungsabläufe leicht auf die Realität übertragen werden können. Die Gewichtung des Vorgehensmodells auf die Erstellung und Gestaltung der Benutzungsoberfläche ist daher ein weiteres wichtiges Bewertungskriterium.

- **Berücksichtigung der Medienproduktion** (kurz: Medienproduktion): Aufgrund der multimedialen Gestaltung der Benutzungsoberfläche sind bei der Entwicklung eines virtuellen Labors eine Reihe von Medien zu erstellen und zu integrieren. So sollte dieses von einem Vorgehensmodell für virtuelle Labore in einer eigenen Phase der Medienproduktion berücksichtigt werden. Ist eine Medienproduktion in dem betrachteten Vorgehensmodell enthalten, so ist zu bewerten, inwiefern diese berücksichtigt wird und wie gut sich die Medienproduktion planen lässt.

3.1.3. Didaktische Aspekte

Ebenso wie die multimedialen Aspekte sind auch die didaktischen Aspekte von einem Vorgehensmodell für virtuelle Labore zu berücksichtigen. Die Erstellung eines didaktischen Konzeptes ist ein wesentlicher Bestandteil bei der Entwicklung virtueller Labore. Es ist der Umfang der zu vermittelnden Lerninhalte festzulegen, eine Strukturierung der Lerninhalte vorzunehmen und eine geeignete Navigationsstruktur auf den Lerninhalten zu bestimmen. Des Weiteren sind die Navigationsmöglichkeiten innerhalb der Lerneinheiten festzulegen. Diesbezüglich ist auch das Interaktionsdesign, also die Wahl der Interaktionsformen und der verwendeten Eingabegeräte, zu berücksichtigen (siehe dazu Boles, 1994, Seite 39 bis 47). Außerdem ist zu ermitteln, wie der Lernerfolg des Lerners im virtuellen Labor gemessen und bewertet werden kann. Es ist also zu bewerten, inwiefern die Erstellung eines didaktischen Konzeptes von dem Vorgehensmodell berücksichtigt wird und welche Bedeutung dem zukommt.

3.1.4. Software-technische Aspekte

Unter den software-technischen Aspekten werden die Adaptierbarkeit des Vorgehensmodells, die Wiederverwendbarkeit bestehender Entwicklungen sowie die Angemessenheit der

Risikoanalysen und der Tests zusammengefasst. Diese Kriterien werden im Folgenden einzeln beschrieben:

- **Adaptierbarkeit des Vorgehensmodells** (kurz: Adaptierbarkeit): Werden die didaktischen oder gestalterischen Anforderungen virtueller Labore nur unzureichend oder gar nicht berücksichtigt, so muss das Vorgehensmodell diesbezüglich angepasst oder erweitert werden. Wichtig ist dabei, dass keine der Anforderungen überbewertet werden. Von dem Vorgehensmodell ist daher eine gleich hohe Gewichtung auf die softwaretechnischen, didaktischen und gestalterischen Aspekte zu legen. Dieses Gleichgewicht im Mittel über den gesamten Software-Entwicklungsprozess zu halten, ist eine der schwierigsten Aufgaben bei der Erstellung virtueller Labore.

- **Wiederverwendbarkeit bestehender Entwicklungen** (kurz: Wiederverw. best. Entw.): Die Entwicklung virtueller Labore stellt eine sehr komplexe und aufwendige Angelegenheit dar und ist aus diesem Grund enorm kostspielig. Von einem Vorgehensmodell für virtuelle Labore sollte daher explizit die Wiederverwendbarkeit bestehender Entwicklungen berücksichtigt werden. Dies betrifft zum einen die Wiederverwendbarkeit bestehender Klassen und Medien, aber insbesondere auch die Wiederverwendung vorhandener Laborgeräte, Behälter und Substanzen sowie die Architektur des virtuellen Labors.

- **Angemessenheit der Tests** (kurz: Angem. der Tests): Beim Testen von virtuellen Laboren ist auf das korrekte Zusammenspiel der multimedialen Elemente von Simulator und intelligentem Tutorsystem zusammen mit den Benutzungsinteraktionen zu achten. Die dafür benötigte Zeit kann kaum reduziert werden, da die Wirkung auf den Lerner nur dann korrekt beurteilt werden kann, wenn sie im zeitlichen Verlauf betrachtet wird. Durch das Testen von virtuellen Laboren entsteht also ein hoher Zeit- und Kostenaufwand, der von dem Vorgehensmodell in einem geeigneten Umfang berücksichtigt werden muss. Die Testphase darf dabei nicht zu kurz sein, damit die Qualität des virtuellen Labors gewährleistet bleibt. Andererseits wird bei einer zu langen Testphase die Entwicklung des virtuellen Labors unwirtschaftlich.

- **Angemessenheit der Risikoanalysen** (kurz: Angem. der Risikoanalysen): Durch Risikoanalysen wird eine gewisse Sicherheit bei der Entwicklung virtueller Labore garantiert. Diese dürfen nicht zu gering ausfallen, da ansonsten keine sinnvolle Einschätzung der Risiken mehr möglich ist. Zu aufwendige Risikoanalysen führen aber dazu, dass die Entwicklung virtueller Labore noch länger dauert und somit noch teurer und unwirtschaftlich wird.

3.1.5. Bewertungsschema

Die in Kapitel 3.1.1 bis 3.1.4 genannten Bewertungskriterien lassen sich in Form eines Bewertungsschemas wie in Tabelle 3.1 darstellen. Die Bewertungsskala zu den Bewertungskriterien geht dabei über sechs Abstufungen von sehr gut bis hin zu ungenügend (nicht vorhanden).

	Bewertungskriterium
Organisatorische Aspekte	Heterogenität des Teams
	Einbez. des Auftraggebers
	Einbez. der Endanwender
Multimediale Aspekte	Benutzungsoberfläche
	Medienproduktion
Didaktische Aspekte	Erstellung der Didaktik
Software-technische Aspekte	Adaptierbarkeit
	Wiederverw. best. Entw.
	Angem. der Tests
	Angem. der Risikoanalysen

Bewertungsskala		
++	=	sehr gut
+	=	gut
o	=	befriedigend
(-)	=	ausreichend
–	=	mangelhaft
n.v.	=	nicht vorhanden

Tabelle 3.1.: Bewertungsschema für Vorgehensmodelle im Hinblick auf den Anwendungsfall virtuelle Labore

3.2. Vorgehensmodelle der klassischen Software-Technik

Die Auswahl der Vorgehensmodelle der klassischen Software-Technik orientiert sich nach (Appelrath u. a., 1998), (Pagel und Six, 1994) und (Balzert, 2000b)[2]. Typische Vertreter von Vorgehensmodellen der klassischen Software-Technik sind demzufolge

- das *Phasenmodell*,
- das *iterierte Phasenmodell*,
- das *Prototypenmodell*,
- die *evolutionäre Software-Entwicklung*,
- die *transformationelle Software-Entwicklung*,
- das *Spiralmodell*,

[2]Das transformationelle Vorgehensmodell und das Spiralmodell sind in diesem Buch nicht enthalten.

- das *V–Modell* und
- die *objektorientierte Software-Entwicklung*.

Diese werden in den Kapiteln 3.2.1 bis 3.2.8 beschrieben. Danach wird in Kapitel 3.2.9 noch das Vorgehensmodell *Catalysis* als eine Erweiterung der objektorientierten Software-Entwicklung vorgestellt. Abschließend wird in Kapitel 3.2.10 eine Bewertung der klassischen Vorgehensmodelle vorgenommen.

3.2.1. Phasenmodell

Das Phasenmodell ist das älteste und bekannteste Vorgehensmodell der klassischen Software-Technik. Der Software-Entwicklungsprozess setzt sich aus den fünf aufeinander folgenden Phasen *Analyse*, *Entwurf*, *Implementierung*, *Test* sowie *Einsatz und Wartung* zusammen (Appelrath u. a., 1998, Seite 108-111). In der Analysephase werden die funktionalen Anforderungen des Auftraggebers an das zu entwickelnde Software-System ermittelt. Es wird die Frage gestellt, was das System leisten soll. Die technische Realisierung ist in dieser Phase nicht von Interesse. In der darauf folgenden Entwurfsphase wird die *System-Spezifikation*, das heißt die Software-Architektur und die Schnittstellen zwischen den Teilen der Software-Architektur festgelegt. In der Implementierung wird die System-Spezifikation in einer konkreten Programmiersprache umgesetzt. Dabei sind unter Umständen die im Entwurf verwendeten Konzepte und Methoden an die Möglichkeiten der Programmiersprache anzupassen. Ergebnis dieser Phase ist ein implementiertes Software-System. In der Testphase werden die implementierten Programmteile auf Vollständigkeit und Korrektheit geprüft und gegebenenfalls Korrekturen an der Implementierung vorgenommen. Ergebnis dieser Phase ist ein ablauffähiges Software-Produkt, das an den Endanwender ausgeliefert werden kann. An die Testphase schließt sich die Phase Einsatz und Wartung an. Diese umfasst alle Aktivitäten, die im Praxiseinsatz des Software-Systems auftreten. Das sind im einzelnen Installation des Software-Systems beim Kunden, Schulung der Benutzer, Korrektur von Fehlern sowie Änderungen und Erweiterungen der Funktionalität nach Wünschen des Kunden. Alle Phasen der Software-Entwicklung werden im Phasenmodell sequentiell durchlaufen. Das bedeutet, dass eine Phase komplett beendet sein muss, bevor mit der nächsten Phase begonnen werden kann. Rücksprünge in frühere Phasen sind im Phasenmodell nicht möglich.

3.2.2. Iteriertes Phasenmodell

Das iterierte Phasenmodell stellt eine Erweiterung des zuvor vorgestellten Phasenmodells dar, bei dem die Rückkehr von der aktuellen Phase in eine frühere Phase mit Hilfe von

Rückkopplungen möglich ist.[3] Im iterierten Phasenmodell kann jeweils von der aktuellen Phase zu einer beliebigen früheren Phase zurückgekehrt werden.

3.2.3. Prototypenmodell

Das Prototypenmodell ist eine Erweiterung des iterierten Phasenmodells um die *Prototyp-Entwicklung*. Dabei wird frühzeitig im Software-Entwicklungsprozess, das heißt während der Analysephase, ein bewusst einfach gehaltener Prototyp erstellt, mit dem der Auftraggeber experimentieren kann, um unklare Anforderungen an das Software-System zu beseitigen. Für die weitere Entwicklung des Software-Produktes wird dieser Prototyp dann nicht mehr verwendet und wird daher auch als *Wegwerf-Prototyp* bezeichnet (vergleiche Zehnder, 1991, Seite 171 bis 173).

3.2.4. Evolutionäre Software-Entwicklung

Bei der evolutionären Software-Entwicklung werden in der initialen *Analysephase* zunächst die Kernanforderungen an das Software-System ermittelt und in einem initialen *Pilotsystem* realisiert. In den darauf folgenden Phasen *Entwurf*, *Implementierung*, *Test* und *Evaluation/Analyse* wird das Pilotsystem iterativ, also aufbauend auf den bereits realisierten Teilen, erweitert. Im Gegensatz zum Wegwerf-Prototypen wird das Pilotsystem also im weiteren Verlauf der Software-Entwicklung weiter verwendet und kontinuierlich verbessert. Der Auftraggeber hat beim evolutionären Modell die Möglichkeit, jeweils Erfahrungen mit der aktuellen Version des Pilotsystems zu sammeln und neue Anforderungen zu definieren. Dies geschieht in der Phase Evaluation/Analyse. In der evolutionären Software-Entwicklung kann jeweils zwischen benachbarten Phasen gewechselt werden. Zudem sind die Phasen zyklisch angeordnet, das heißt es kann zusätzlich noch zwischen der Phase Entwurf und Evaluation/Analyse gewechselt werden. Entspricht das Pilotsystem den Wünschen des Auftraggebers, so ist die Entwicklung zum endgültigen Software-Produkt abgeschlossen.

3.2.5. Transformationelle Software-Entwicklung

Bei der transformationellen Software-Entwicklung werden die funktionalen Anforderungen an das Software-System in einer formalen System-Spezifikation definiert. Die formale System-Spezifikation wird dann in einem automatisierten Prozess in die Programmiersprache, also das Software-Produkt transformiert (vergleiche das AVALON-System für die transformationelle Software-Entwicklung von Multimedia-Systemen in Boles und Wütherich, 1997). In diesem

[3] Achtung: In der Literatur wird das iterierte Phasenmodell oft auch als Wasserfallmodell bezeichnet, wie zum Beispiel in (Sawhney, 1995, Seite 16), (Yass, 2000, Seite 142) und (Frühauf u. a., 1991b, Seite 15). Unter dem Wasserfallmodell wird in dieser Arbeit jedoch das klassische Phasenmodell verstanden (wie auch in Appelrath u. a., 1998; Balzert, 2000a; Pagel und Six, 1994).

Zusammenhang wird auch von einer *Generierung des Software-Systems* gesprochen.[4] Realistischer ist allerdings der Ansatz, bei der die Transformation von der Spezifikation zum Software-Produkt nur halb-automatisch unter Verwendung eines wissensbasierten Systems erfolgt (siehe auch Pagel und Six, 1994, Seite 69).

3.2.6. Spiralmodell

Das Spiralmodell ist ein Metamodell, in das andere Vorgehensmodelle eingebettet werden. Ziel des Spiralmodells ist die Risikominimierung. Der Software-Entwicklungsprozess wird beim Spiralmodell als iterativer Prozess angesehen und durch mehrfaches Durchlaufen der Spirale realisiert. Die Iterationen werden im Spiralmodell *Windungen* genannt. Zu Beginn jeder Windung wird geplant, welche Aktivitäten durchzuführen sind und welches Vorgehensmodell verwendet werden soll. Außerdem werden die Ziele formuliert, die mit dieser Windung erreicht werden sollen. Anschließend wird das ausgewählte Vorgehensmodell angewendet. Am Ende wird das Ergebnis der Windung mit den zuvor gesteckten Zielen überprüft und bewertet. Ist das Risiko für eine Fortführung des Projektes zu groß, so wird die Software-Entwicklung abgebrochen. Entspricht das Ergebnis den Anforderungen des Kunden, so ist die Entwicklung zum fertigen Software-Produkt abgeschlossen.

3.2.7. V–Modell

Das V–Modell beschreibt den Software-Entwicklungsprozess mit Hilfe von *Aktivitäten* und *Produkten*[5] (siehe IABG, 2001; ANSSTAND e.V., 2001). Außerdem legt das V–Modell fest, mit welchen Methoden die Aktivitäten durchzuführen und mit welchen Mitteln die Artefakte darzustellen sind. Des Weiteren werden im V–Modell die funktionalen Anforderungen der zum Einsatz kommenden Werkzeuge festgelegt. Neben der Software-Entwicklung im engeren Sinne wird im V–Modell ein weiterer Schwerpunkt auf das Qualitäts-, Konfigurations- und Projektmanagement gelegt. So nimmt das Testen des Software-Systems beim V–Modell einen sehr hohen Stellenwert ein. Neben dem militärischen Bereich ist das V–Modell auch für den gesamten Bereich der Bundesverwaltung verbindlich und wird von sehr vielen Industriefirmen als Hausstandard zur Software-Entwicklung verwendet. Mit Hilfe des V–Modells können Software-Projekte gemäß der Norm ISO 9001 abgewickelt werden.

3.2.8. Objektorientierte Software-Entwicklung

Bei der objektorientierten Software-Entwicklung werden die Ergebnisse der Phasen *Analyse*, *Entwurf* und *Implementierung* objektorientiert erstellt. Dazu werden bei der *objektorientierten Analyse* (OOA) die Anforderungen an das Software-System mittels

[4] Eine automatisierte Generierung von Software-Systemen ist zum Beispiel mit dem Programmgenerator *JANUS* der Firma *oTRIs* (siehe: http://www.otris.de/) möglich (siehe Balzert, 2000a, Seite 15).
[5] Die im V–Modell genannten Artefakte.

objektorientierter Konzepte und Notationen ermittelt und beschrieben. Das wichtigste Ergebnis der Analysephase ist das *OOA-Modell*, das eine fachliche Lösung des zu realisierenden Systems darstellt (vergleiche Analyse des Anwendungsgebietes in Appelrath u. a., 1998, Seite 126). Nach der objektorientierten Analyse folgt der *objektorientierte Entwurf* (*object oriented design*, OOD). Aufbauend auf dem OOA-Modell wird bei dem objektorientierten Entwurf die Software-Architektur und die Spezifikation der Klassen aus Sicht der Realisierung erstellt. Ergebnis des objektorientierten Entwurfs ist das *OOD-Modell*, das die technische Lösung des zu realisierenden Systems darstellt. Das OOD-Modell ist ein Abbild des objektorientierten Programms auf einem höheren Abstraktionsniveau. In der Implementierung, der *objektorientierten Programmierung* (OOP), wird das OOD-Modell auf die Konzepte und Notationen der verwendeten objektorientierten Programmiersprache abgebildet. Im Idealfall lassen sich sämtliche im OOD-Modell verwendeten Konzepte und Notationen direkt mit den Konzepten und Notationen der objektorientierten Programmiersprache darstellen. Bei der objektorientierten Software-Entwicklung steht vor allem die *Wiederverwendung* im Vordergrund und kann auf den Ebenen der objektorientierten Analyse, des objektorientierten Entwurfs und der objektorientierten Programmierung erfolgen. Für den Test objektorientierter Software (*objektorientierter Test*, kurz: OOT) können die traditionellen Testmethoden nahezu unverändert übernommen werden. Es werden zunächst die einzelnen Klassen für sich und dann ihr Zusammenspiel getestet. Die Grenzen zwischen den einzelnen Phasen der objektorientierten Software-Entwicklung sind im Gegensatz zum Phasenmodell fließend.

3.2.9. Catalysis

Das Vorgehensmodell Catalysis ist eine Erweiterung der objektorientierten Software-Entwicklung um einen auf Komponenten basierenden objektorientierten Entwurf (D'Souza und Wills, 1998). Die Software-Entwicklung findet bei Catalysis auf drei Modellierungsebenen statt. Auf der *Geschäftsmodellebene* werden die Anforderungen an das Software-System unabhängig von der eingesetzten Entwicklungsumgebung (*CASE-Umgebung*) festgehalten. Das Verhalten der einzelnen Komponenten des Software-Systems und das gemeinsame Zusammenspiel aller Komponenten wird auf der Ebene des *Komponentenentwurfs* beschrieben. Des Weiteren findet für jede Komponente ein *Objektentwurf* statt. Im Objektentwurf wird die Arbeitsweise einer Komponente mit Hilfe von Klassen der verwendeten Programmiersprache im Detail beschrieben. Catalysis legt besonderen Wert auf die Konsistenzerhaltung und Nachvollziehbarkeit der Entwurfsdokumente, da ansonsten Inkonsistenzen zwischen den verschiedenen Modellierungsebenen entstehen. Zur Beschreibung der Elemente von Catalysis, wie zum Beispiel Klassen und Komponenten, wird die Unified Modeling Language verwendet. Catalysis hat durch den Einfluss auf die *Object Constraint Language* (OCL) auch eine indirekte Auswirkung auf die Entwicklung der Unified Modeling Language. Die Object Constraint Language ist der Beitrag von IBM für die Unified Modeling Language und wurde zur Spezifikation der Semantik der Unified Modeling Language entwickelt (siehe Object Management Group, 1997).

3.2.10. Bewertung der klassischen Vorgehensmodelle

In (Sawhney, 1995) werden die wichtigsten Vorgehensmodelle der klassischen Software-Technik analysiert, und es wird untersucht, inwiefern diese Vorgehensmodelle für die Entwicklung von Multimedia-Systemen geeignet sind. Danach eignet sich das reine Phasenmodell nicht für die Entwicklung von Multimedia-Systemen, da diese eine sehr hohe Gewichtung auf die Gestaltung der Benutzungsoberfläche legen. Aus diesem Grund ist das Phasenmodell auch nicht für die Entwicklung von virtuellen Laboren einsetzbar (siehe Boles, 1997, Seite 3). Durch das Fehlen von Rücksprüngen zu früheren Phasen der Software-Entwicklung ist das Phasenmodell außerdem für die Entwicklung von Software-Systemen ungeeignet, bei denen flexibel auf die sich ändernden Anforderungen reagiert werden muss (Hruschka u. a., 2001, Seite 21), wie zum Beispiel bei der Neuentwicklung von virtuellen Laboren.

Das iterierte Phasenmodell ist mit seinen Rückkopplungen zu früheren Phasen zwar eine wesentliche Verbesserung gegenüber dem Phasenmodell, da der Auftraggeber jedoch wie beim Phasenmodell nur in der Analysephase an der Software-Entwicklung beteiligt ist, sind Änderungswünsche an der Benutzungsoberfläche und den Funktionalitäten erst in der Phase Einsatz/Wartung möglich. Eine Korrektur des Software-Systems kann also nur sehr spät im Software-Entwicklungsprozess erfolgen und wird damit sehr teuer.

Auftraggeber sind jedoch häufig nicht in der Lage, zu Beginn eines Software-Projektes die gewünschten Anforderungen an das Software-Produkt präzise zu formulieren (Appelrath u. a., 1998, Seite 109). Das gilt insbesondere für interaktive Software-Systeme (Sawhney, 1995, Seite 17), wie zum Beispiel virtuelle Labore. Daher ist eher das Prototypenmodell oder die evolutionäre Software-Entwicklung mit häufiger Evaluation durch den Auftraggeber und Endanwender zu wählen (vergleiche die Bewertung des evolutionären Modells in Balzert, 2000b, Seite 57). Im Prototypenmodell spielt nicht die Qualität des Prototypen, sondern die Funktionalität, Veränderbarkeit und schnelle Entwicklung eine Rolle (vergleiche exploratives Prototyping in Sawhney, 1995, Seite 18). Aufwendigere Prototypen zur Demonstration der Gestaltung der Benutzungsoberfläche kommen wegen der hohen Kosten zur Erstellung der Benutzungsoberfläche nicht in Frage, zumal der entwickelte Prototyp nach der Evaluation nicht weiter verwendet wird. Dem gegenüber sieht die evolutionäre Software-Entwicklung eine kontinuierliche Entwicklung des Pilotsystems vor (vergleiche evolutionäres Prototyping in Sawhney, 1995, Seite 18). Aber auch dieses Vorgehensmodell ist wegen der Medienproduktion wenig geeignet, da diese durch Einbeziehung von externen Firmen oder der Anmietung von Studios genauestens geplant sein muss (siehe Sawhney, 1995, Seite 32).

Das transformationelle Vorgehensmodell kann nicht für die Entwicklung von virtuellen Laboren eingesetzt werden, da für diesen Anwendungsbereich keine vollständige formale Spezifikation durchgeführt werden kann. So können zum Beispiel die gestalterischen und didaktischen Aspekte nicht formal erfasst werden. Für den Simulator eines virtuellen Labors ist das transformationelle Vorgehensmodell jedoch sehr gut geeignet, da dieser strengen naturwissenschaftlichen Gesetzen gehorcht. Liegen die Versuchsabläufe in einer formalen Spezifikationssprache vor, so kann das transformationelle Vorgehensmodell auch zur automatisierten Generierung von Versuchsanleitungen eingesetzt werden. Transformationelle

Vorgehensmodelle werden allerdings in der industriellen Praxis bisher nur unzureichend unterstützt und nur in wenigen Bereichen, wie zum Beispiel für sicherheitskritische eingebettete Systeme, auch tatsächlich eingesetzt.

Der Einsatz des Spiralmodells wird abgelehnt, weil aufgrund der sehr aufwendigen Risikoanalysen nach jeder Windung die Wirtschaftlichkeit des Software-Projektes nicht garantiert werden kann. Außerdem ist das Spiralmodell nach (Hruschka u. a., 2001) wegen der nur einmal vorhandenen Analysephase weniger flexibel, als zum Beispiel das iterative Vorgehensmodell.

Das V–Modell legt einen Schwerpunkt auf Validation und Verifikation des zu erstellenden Software-Systems. Da das Testen virtueller Labore sehr aufwendig ist und kaum verkürzt werden kann, dauert eine Entwicklung nach dem V–Modell entsprechend lange und ist somit sehr teuer. Der Vorteil des V–Modells ist jedoch das bereits vorhandene Rollenkonzept, dass nur noch auf die Rollen zur Entwicklung virtueller Labore angepasst werden muss.

Einer der größten Vorteile der objektorientierten Software-Entwicklung ist die perfekte Durchgängigkeit von der objektorientierten Analyse, über den objektorientierten Entwurf bis zur objektorientierten Implementierung (Balzert, 2000a, Seite 168). Dieser Vorteil wird zum Beispiel unter Verwendung der Unified Modeling Language als Modellierungssprache für die objektorientierte Analyse und den objektorientierten Entwurf und zum Beispiel Java als objektorientierte Programmiersprache erreicht. Mit den Methoden zur Klassenbildung und Vererbung ist die objektorientierte Software-Entwicklung sehr gut zur Modellierung der Domäne des virtuellen Labors und insbesondere zur Erstellung des Simulators geeignet. Bei auftretenden Fehlern oder späteren Erweiterungen des Software-Systems kommen die weiteren Vorteile der objektorientierten Software-Entwicklung wie Datenkapselung, Erweiterbarkeit und Wiederverwendbarkeit zum Tragen. Vor allem die Wiederverwendbarkeit kommt dem Aspekt einer kostengünstigen Produktion und dem Aufbau beziehungsweise Einsatz eines Frameworks für virtuelle Labore sehr entgegen (siehe dazu Hasler und Schlattmann, 2001, Seite 23).

Nach (Balzert, 2000b, Seite 856) erlaubt eine komponentenbasierte Software-Entwicklung eine einfache, schnelle und preiswerte Herstellung von Software-Systemen mit Hilfe von vorgefertigten Komponenten. Damit unterstützt das Vorgehensmodell Catalysis wesentlich das von *VirtLab* erklärte Ziel einer schnellen und kostengünstigen Produktion qualitativ hochwertiger virtueller Labore. Dabei ist jeweils im konkreten Fall zu prüfen, inwiefern die komponentenbasierte Software-Entwicklung notwendig ist, oder ob nicht bereits eine Aufteilung des Software-Systems in mehrere Teilsysteme ausreicht. Die objektorientierte Software-Entwicklung und Catalysis werden der Komplexität virtueller Labore jedoch nur aus software-technischer Sicht gerecht. Bei beiden Vorgehensmodellen bleiben die gestalterischen und didaktischen Aspekte unberücksichtigt. Damit sind auch diese Vorgehensmodelle nur bedingt für die Entwicklung virtueller Labore anwendbar.

Zusammenfassend lässt sich sagen, dass keines der klassischen Vorgehensmodelle für die Entwicklung virtueller Labore befriedigend ist. Die wesentliche Ursache dafür ist in der fehlenden Berücksichtigung der multimedialen und didaktischen Aspekte virtueller Labore zu sehen. Des Weiteren werden nur von dem V–Modell und dem Spiralmodell (bei Einbettung

eines entsprechenden Vorgehensmodells) die Heterogenität des Entwicklerteams durch ein entsprechendes Rollenkonzept auch tatsächlich berücksichtigt.

3.3. Spezielle Vorgehensmodelle

Neben den bisher genannten Vorgehensmodellen aus der klassischen Software-Technik existieren in der Literatur auch zahlreiche Vorgehensmodelle, die speziell auf die Entwicklung von Multimedia-Systemen, Lehr- und Lernsystemen und multimedialen Lehr- und Lernsystemen zugeschnitten sind. Diese wurden entwickelt, da sich aus den Multimedia-Systemen und Lehr- und Lernsystemen neue Anforderungen an die Software-Entwicklung, wie zum Beispiel die Medienproduktion und der Aufbau der Didaktik, ergeben haben. Für die Entwicklung von Multimedia-Systemen wird in Kapitel 3.3.1 das *Vorgehensmodell für die Multimedia-Anwendungsentwicklung* (nach Sawhney, 1995) vorgestellt. Als Beispiel für die Entwicklung von Lehr- und Lernsystemen wird in Kapitel 3.3.2 das *Vorgehensmodell zur Entwicklung von Lehr- und Lernsystemen* (nach Yass, 2000) beschrieben. Für die Entwicklung von multimedialen Lehr- und Lernsystemen wird in Kapitel 3.3.3 das *Vorgehensmodell für die industrielle Entwicklung multimedialer Lehr- und Lernsysteme* (nach Nagl u. a., 1999) betrachtet. Abschließend erfolgt in Kapitel 3.3.4 eine Bewertung der speziellen Vorgehensmodelle.

3.3.1. Vorgehensmodell für die Multimedia-Anwendungsentwicklung

Bei dem in (Sawhney, 1995) beschriebenen Vorgehensmodell für die Multimedia-Anwendungsentwicklung stehen die Gestaltungsaspekte des zu entwickelnden Multimedia-Systems und nicht die zu implementierenden Funktionalitäten im Vordergrund.

Wie aus Abbildung 3.1 ersichtlich, teilt sich das Vorgehensmodell in die Phasen *Analyse*, *System-Spezifikation*, *Konzeption*, die parallel ablaufenden Phasen *Programmrealisation* und *Medienproduktion*, sowie die darauf folgenden Phasen *System-Test* und *Wartung und Pflege*. Die Analysephase teilt sich in die Benutzer-, System-, Umgebungs- und Inhaltsanalyse auf. Danach erfolgt die System-Spezifikation. In der Konzeptionsphase wird basierend auf die System-Spezifikation die Navigationsstruktur und die Benutzungsoberfläche des Multimedia-Systems festgelegt. Die technische Umsetzung des Multimedia-Systems erfolgt in der Programmrealisation und der Medienproduktion. Zur Entwicklung des Multimedia-Systems wird die *Top-Down*-Strategie vorgeschlagen, das heißt es werden Entscheidungen auf höherer Ebene getroffen und dann schrittweise verfeinert. Nach der Analysephase vollzieht sich die Entwicklung des Multimedia-Systems bis zum Beginn der Programmrealisation in einem evolutionären Prozess. Aufgrund der hohen Kosten, die durch die Medienproduktion entstehen, sind nach der Fertigstellung und Validierung des *Drehbuchs* (*script*) keine Phasenwiederholungen möglich. Daher findet im Vorgehensmodell ab der Programmrealisation und Medienproduktion eine strenge Phaseneinhaltung statt.

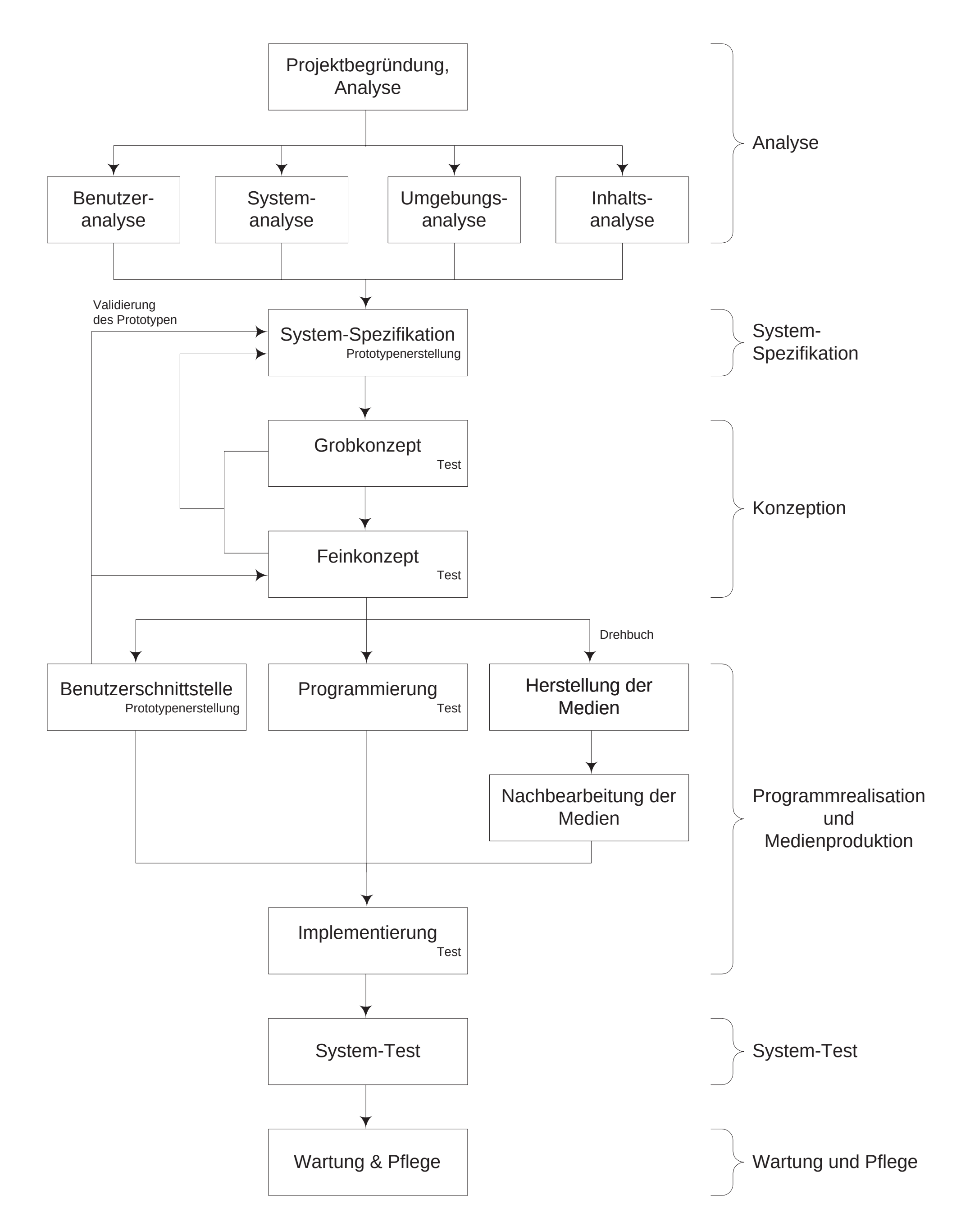


Abbildung 3.1.: Vorgehensmodell für die Multimedia-Anwendungsentwicklung (nach Sawhney, 1995)

In Abbildung 3.2 ist die *Medienherstellung* dargestellt. Diese beginnt bereits in der Analysephase mit der *Medienanalyse*, wird in der Konzeptphase mit der *Grob- und Feinplanung des Inhalts* fortgesetzt und endet in der Medienproduktionsphase mit der *Herstellung und Nachbearbeitung der Medien*. In den Analyse- und Konzeptionsphasen sind Rücksprünge zu vorhergehenden Phasen erlaubt. Nach der Feinplanung des Inhalts,

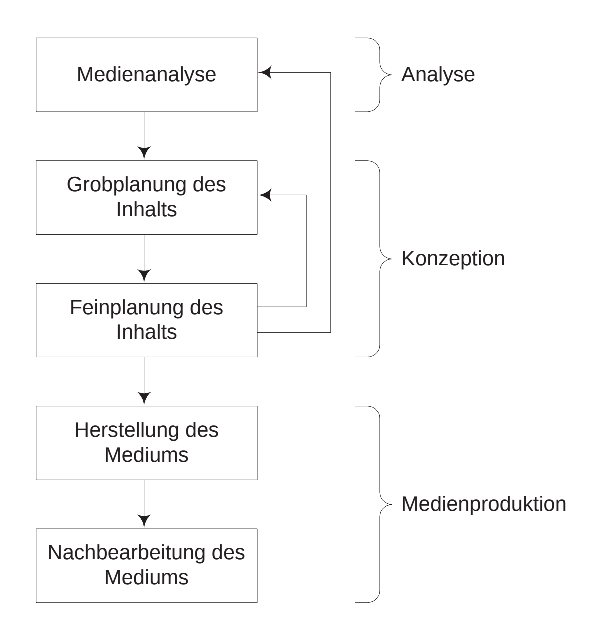


Abbildung 3.2.: Ablauf der Medienherstellung (nach Sawhney, 1995)

also zu Beginn der Medienproduktion, sind Rücksprünge in der Medienherstellung aus Kostengründen nicht mehr möglich.

3.3.2. Vorgehensmodell zur Entwicklung von Lehr- und Lernsystemen

In (Yass, 2000, Seite 296 ff.) wird ein Vorgehensmodell zur Entwicklung von Lehr- und Lernsystemen vorgestellt. Nach diesem Vorgehensmodell teilt sich der Software-Entwicklungsprozess grob in die Phasen *Analyse*, *Projektdefinition*, *Design*, *Entwicklung*, *Test und Auswertung*, *Distribution und Einführung* und *Wartung* auf. Diese können teilweise noch weiter untergliedert werden. In der Analysephase wird geklärt, ob ein Lehr- und Lernsystem zu dem vorliegenden Thema sinnvoll ist. Dazu wird eine Bedarfsanalyse durchgeführt und der voraussichtliche Zeit- und Kostenaufwand des zu erstellenden Software-Systems geschätzt. In der Projektdefinitionsphase werden *Lernziele* und *Lerneinheiten* spezifiziert und die Didaktik der Lerneinheiten aufgebaut. Zudem werden Richtlinien für ein einheitliches Bildschirmlayout und die Navigationsstruktur festgelegt (vergleiche auch das IMRA-Modell in Boles, 1994, Seite 85-122). In der darauf folgenden Designphase wird das *Regiebuch* erstellt. Im Regiebuch werden unter anderem die Lerninhalte und eine Liste der benötigten Medien festgehalten. Die Lerninhalte werden basierend auf den Lernzielen, die in der Definitionsphase ausgearbeitet worden sind, ausgewählt und vorbereitet. Bei den einzelnen Medien wird entschieden, ob sie erstellt oder beschafft werden. Außerdem beinhaltet das Regiebuch genaue Arbeitsanweisungen für die Entwickler und den Ablaufplan des Lehr- und Lernsystems. Um die Entwicklungskosten durch spätere Änderungen nicht unnötig in die Höhe zu treiben, wird empfohlen das Regiebuch abschließend mit dem Auftraggeber zu besprechen. In der Entwicklungsphase werden die Anweisungen und Ergebnisse des

Regiebuchs in ein konkretes Programm umgesetzt. Dabei wird ein prototypisches oder evolutionäres Vorgehen zur Software-Entwicklung vorgeschlagen. Spätestens in dieser Phase muss die endgültige Entscheidung für ein Autorensystem oder eine Programmiersprache mit entsprechenden Programmbibliotheken gefallen sein. Ergebnis der Entwicklungsphase ist ein ablauffähiges Software-System, sowie die zu erstellende Dokumentation. In der Test- und Auswertungsphase wird das Lehr- und Lernsystem neben Fehlern in der Implementierung auch auf didaktische und inhaltliche Fehler, gestalterische Mängel sowie Abweichungen von den Lernzielen geprüft und korrigiert. In der Phase Distribution und Einführung wird der Einsatz des Software-Systems vorbereitet. In der abschließenden Wartungsphase werden vom Kunden entdeckte Fehler und Mängel an dem Lehr- und Lernsystem behoben, sowie von ihm geäußerte Änderungswünsche durchgeführt.

3.3.3. Vorgehensmodell für die industrielle Entwicklung multimedialer Lehr- und Lernsysteme

In der Studie von (Nagl u. a., 1999) über die software-technischen Anforderungen an multimediale Lehr- und Lernsysteme wird ein Vorgehensmodell für die industrielle Entwicklung multimedialer Lehr- und Lernsysteme beschrieben. Nach der klassischen Software-Technik setzt sich die Software-Entwicklung aus den drei interagierenden Bereichen *Software-Management*, *Software-Entwicklung* und *Software-Qualitätssicherung* zusammen (siehe Balzert, 2000b). Diese stellen im Wesentlichen die Grundlage der Multimedia-Entwicklung dar.

Das Hauptziel des Software-Managements ist die erfolgreiche Durchführung einer Software-Entwicklung. Die Aufgaben des Software-Managements sind die Einsatzmittelplanung, also die Termin-, Ressourcen- und Kostenplanung, sowie die Koordination der Einzelaufgaben innerhalb der Gesamtentwicklung. Für die Entwicklung von multimedialen Lehr- und Lernsystemen wird ein lineares Vorgehensmodell vorgeschlagen (siehe Abbildung 3.3 aus Nagl u. a., 1999, Seite 117). Dieses ergibt sich aus den industriellen Randbedingungen der Budgetierung und Terminierung der Software-Entwicklung. Können diese Randbedingungen vernachlässigt werden, entwickelt sich das Vorgehensmodell in Richtung des evolutionären Modells, in dem es mehr Rückkopplungen gibt.

Die Software-Entwicklung besteht aus den sechs aufeinander aufbauenden Phasen *Planung*, *Definition*, *Entwurf*, *Implementierung*, *Abnahme und Einführung* sowie *Wartung und Pflege*. Diese entsprechen im Wesentlichen den Phasen des Phasenmodells. Allerdings umfasst die Analysephase des Phasenmodells bereits die beiden Phasen Planung und Definition. Die Testphase wird hier nicht als eigenständige Phase angenommen. Die Phase Einsatz und Wartung aus dem Phasenmodell deckt die beiden Phasen Abnahme und Einführung sowie Wartung und Pflege ab. Bei der Entwicklung von multimedialen Lehr- und Lernsystemen kommt parallel zur Implementierungsphase noch die Medienproduktionsphase hinzu, die die *Medienherstellung* sowie den *Medientest* beinhaltet. Ergebnis der Planungsphase ist das

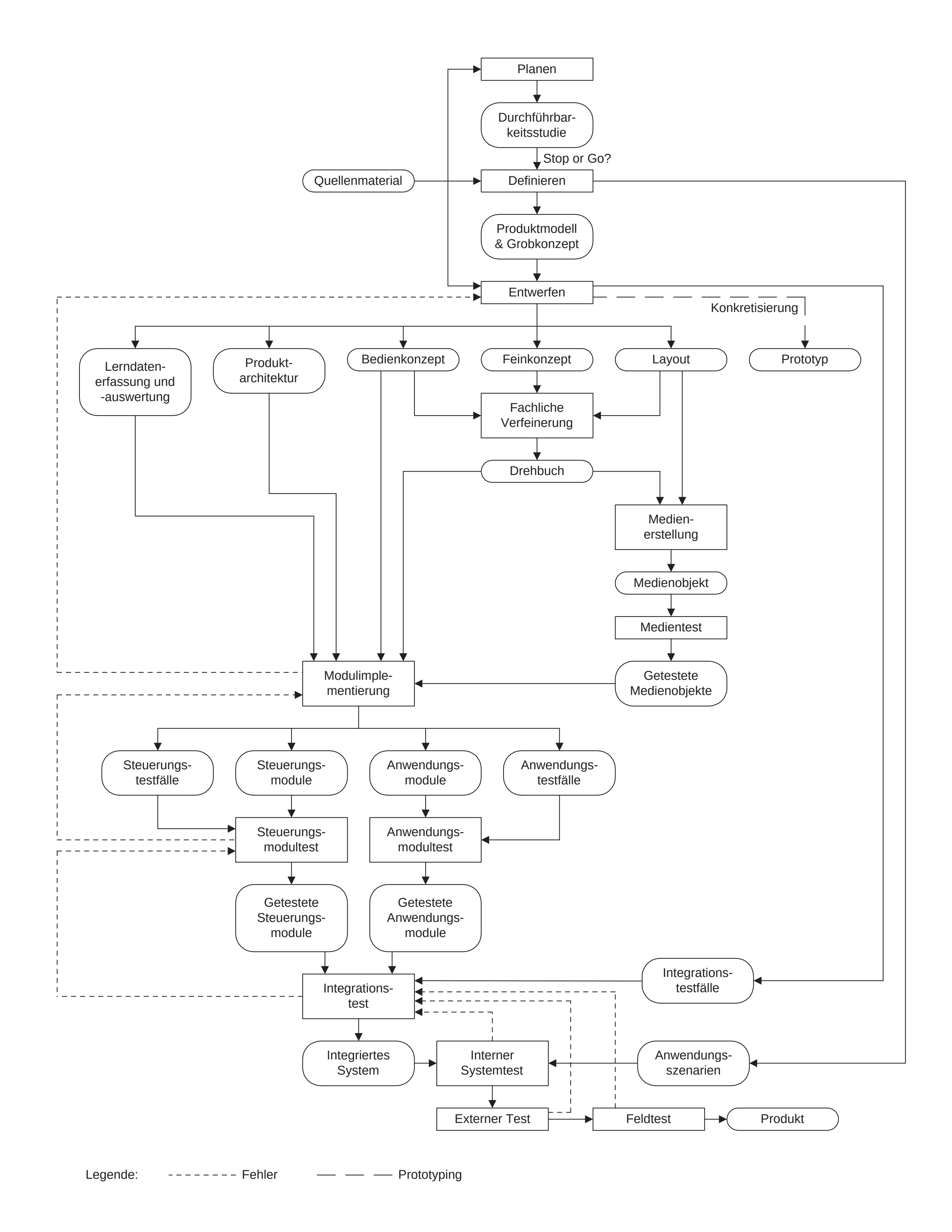


Abbildung 3.3.: Tätigkeiten und Dokumente bei der Multimedia-Entwicklung (nach Nagl u. a., 1999)

Lastenheft[6] und kommt noch vor der Anforderungsdefinition, die in der Definitionsphase erstellt wird.

Unter Software-Qualität versteht man nach DIN ISO 9126 die Gesamtheit der Merkmale und Merkmalswerte eines Software-Produktes, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen (siehe Nagl u. a., 1999, Seite 142). Bei multimedialen Lehr- und Lernsystemen fließen neben den klassischen software-technischen Merkmalen auch inhaltliche und mediendidaktische Qualitätsmerkmale ein. Im Gegensatz zu den meisten klassischen Software-Systemen unterliegen multimediale Lehr- und Lernsysteme der Dynamik und Zeitbehaftung. Videosequenzen und Animationen haben eine gewisse Laufzeit, die kaum reduziert werden kann. Das Testen von multimedialen Lehr- und Lernsystemen ist dadurch mit einem höheren Zeitaufwand und damit mit höheren Kosten verbunden.

3.3.4. Bewertung der speziellen Vorgehensmodelle

Bei dem Vorgehensmodell für die Multimedia-Anwendungsentwicklung nach (Sawhney, 1995) handelt es sich um eine Kombination der evolutionären Software-Entwicklung und des Phasenmodells, erweitert um Aspekte für die Entwicklung von Multimedia-Systemen. Bis zur Konzeptionsphase vollzieht sich die Erstellung des Multimedia-Systems in einem evolutionären Prozess. Der Vorteil der evolutionären Software-Entwicklung den Auftraggeber möglichst frühzeitig und häufig in den Software-Entwicklungsprozess einzubeziehen, ist damit auch in dem speziellen Vorgehensmodell gegeben. Der Nachteil der evolutionären Software-Entwicklung bezüglich der nur unzureichend planbaren Medienproduktion wird durch die strikte Phaseneinhaltung ab der Konzeptionsphase behoben. Für die Entwicklung von virtuellen Laboren ist das Vorgehensmodell allerdings weniger geeignet, da es sich eher in Richtung hypermediale Software-Systeme orientiert (vergleiche Sawhney, 1995, Seite 42 bis 44). Die hohe Interaktivität und die didaktischen Anforderungen virtueller Labore werden von diesem Vorgehensmodell nicht berücksichtigt.

Das Vorgehensmodell zur Entwicklung von Lehr- und Lernsystemen nach (Yass, 2000) ist eine logische Erweiterung des Phasenmodells um die Aspekte zur Entwicklung von Lehr- und Lernsystemen. So wird das Vorgehensmodell der Erstellung eines didaktischen Konzeptes für Lehr- und Lernsysteme gerecht. Da das Vorgehensmodell aber eher auf die Entwicklung von klassischen Lehr- und Lernsystemen nach Kapitel 2.4 ausgerichtet ist (siehe Yass, 2000, Seite 280 ff.), eignet es sich weniger für die Entwicklung von virtuellen Laboren.

Das in (Nagl u. a., 1999) beschriebene Vorgehensmodell für die industrielle Entwicklung multimedialer Lehr- und Lernsysteme ist eine konsequente Weiterentwicklung der Elemente der klassischen Software-Technik bezüglich der Anforderungen, die sich seitens der multimedialen Lehr- und Lernsysteme ergeben. Aber auch dieses spezielle Vorgehensmodell dient eher zur Entwicklung von klassischen Lehr- und Lernsystemen. Des Weiteren lassen sich die Aktivitäten und Artefakte bei der Entwicklung von multimedialen Lehr- und Lernsystemen nicht mehr besonders übersichtlich darstellen (siehe Abbildung 3.3).

[6] Auch: *grobes Pflichtenheft* (siehe Balzert, 2000b, Seite 62)

Insgesamt lässt sich feststellen, dass die speziellen Vorgehensmodelle eher auf die Entwicklung klassischer, hypermedialer Lehr- und Lernsysteme zugeschnitten sind und sich somit nicht unbedingt für die Entwicklung virtueller Labore eignen. Gegenüber den klassischen Vorgehensmodellen stellen die speziellen Vorgehensmodelle allerdings eine wesentliche Verbesserung dar, da gestalterische und didaktische Aspekte berücksichtigt werden.

3.4. Vorgehensmodelle der modernen Software-Technik

Die in diesem Kapitel betrachteten Vorgehensmodelle stellen im gewissen Sinne eine neue Vorgehensweise in der Software-Technik dar. Viele Konzepte und Methoden der bisher genannten Vorgehensmodelle finden sich hier wieder oder wurden weiterentwickelt (vergleiche Jacobson u. a., 1999; Kruchten, 2000; oose.de GmbH, 1999; Beck, 2000). In Kapitel 3.4.1 werden zunächst einige moderne Vorgehensmodelle beschrieben, die auf Workflows basieren. Das sind konkret

- der *Unified Software Development Process,*
- der *Rational Unified Process* und
- der *Object Engineering Process*.

Durch die hohe Anzahl der zu erstellenden Artefakte werden diese Vorgehensmodelle auch als *schwergewichtige Vorgehensmodelle* bezeichnet (Coldewey, 2001, Seite 23). Als der zur Zeit populärste Vertreter der *leichtgewichtigen Vorgehensmodelle*, wird abschließend noch in Kapitel 3.4.2 das Vorgehensmodell des *Extreme Programming* betrachtet (Dittmar und Eckstein, 2001, Seite 28). Danach erfolgt in Kapitel 3.4.3 die Bewertung der modernen Vorgehensmodelle.

3.4.1. Auf Workflows basierende Vorgehensmodelle (USDP, RUP, OEP)

Die Definition und Koordination der statischen Prozessstruktur erfolgt bei den auf Workflows basierenden Vorgehensmodellen wie der Unified Software Development Process (USDP), Rational Unified Process (RUP) und der Object Engineering Process (OEP) mit Hilfe von *Workflows*[7]. Die Anzahl der verwendeten Workflows variiert bei den Vorgehensmodellen und liegt typischerweise zwischen fünf und neun. Jedes Vorgehensmodell legt bestimmte Workflows für die Software-Entwicklung fest. Die Elemente der Workflows sind *Rollen*, *Aktivitäten* und *Artefakte*. Einer Rolle sind Aktivitäten und Artefakte zugeordnet. Eine

[7] Auch: Kernprozesse (siehe Balzert, 2000a, Seite 224)

Aktivität stellt eine Arbeitseinheit dar, die von einem Mitarbeiter, der sich in einer bestimmten Rolle befindet, durchgeführt wird (*Rollenkonzept*). Es erfolgt also eine Trennung zwischen Menschen und den Rollen, die die Menschen innehaben können (vergleiche funktionelle Rolle in Pasch, 1994, Seite 171 und Seite 180). Das Ergebnis einer Aktivität ist ein Artefakt. Ein Artefakt kann ein Dokument (zum Beispiel eine Risikoliste), ein Modell (zum Beispiel ein Anwendungsfallmodell) oder ein Modellelement (zum Beispiel eine Klasse eines Klassenmodells) sein. Es existieren Leitlinien (*Guidelines*) und Schablonen (*Templates*), anhand dessen die Aktivitäten durchgeführt werden können.

Weiteres gemeinsames Kennzeichen der auf Workflows basierenden Vorgehensmodelle ist, dass der Software-Entwicklungsprozess hauptsächlich durch *Anwendungsfälle* (*use-cases*) vorangetrieben wird (Jacobson u. a., 1999, Seite 33 ff.). *Anwendungsfallgetrieben* bedeutet, dass zur Erhebung der Anforderungen an das Software-System hauptsächlich Anwendungsfälle eingesetzt werden. Durch einen Anwendungsfall wird ein typisches Szenario des zu entwickelnden Software-Systems beschrieben, wie zum Beispiel die Aufnahme eines neuen Kunden oder einer neuen Bestellung. Anwendungsfälle ermöglichen dem Benutzer beziehungsweise Auftraggeber über die Funktionalität des Software-Systems zu sprechen, ohne sich in Details zu verlieren. Sie werden am Anfang systematisch erarbeitet und bilden die Grundlage für das weitere Vorgehen.

Die Vorgehensmodelle werden zudem als *architekturzentriert* bezeichnet, weil sie nach den Eigenschaften und Besonderheiten einer vorhandenen Anwendungsarchitektur ausgerichtet werden können (Jacobson u. a., 1999, Seite 59 ff.). Architekturen sind häufig Schichtenmodelle und beinhalten beispielsweise bei der *Drei–Schichten–Architektur* die Ebenen *Benutzungsoberfläche*, *eigentliche Anwendung* und *Datenhaltung* (siehe Balzert, 2000b, Seite 716). Oft werden Schichten-Architekturen im Zusammenhang mit *Client/Server*-Anwendungen betrachtet (vergleiche Tanenbaum, 1996, Seite 3 f.). Dabei wird die Verteilung des Software-Systems auf Server, die bestimmte Dienste zur Verfügung stellen, und Clients, die Dienste der Server in Anspruch nehmen, betrachtet.

Ein weiteres gemeinsames Merkmal ist die *iterative* und *inkrementelle* Durchführung der Software-Entwicklung (siehe Jacobson u. a., 1999, Seite 85 ff.). Iterativ bedeutet, dass die Software-Entwicklung in mehrere gleichartige Schritte (*Iterationen*) zerlegt wird. Jede Iteration besteht aus einer Feinplanung, in der die Iterationsziele festgelegt werden, und einer Abfolge der Workflows des Vorgehensmodells. In jeder Iteration wird ein ablauffähiges Pilotsystem erzeugt, das getestet und weiterentwickelt werden kann. Inkrementell bedeutet, dass die Gesamtfunktionalität des zu entwickelnden Software-Systems mit jedem Schritt zunimmt (siehe die evolutionäre Software-Entwicklung in Kapitel 3.2.4).

Der komplette Software-Entwicklungsprozess erstreckt sich bei den auf Workflows basierenden Vorgehensmodellen über mehrere nacheinander ablaufende Phasen, die dem Ablauf der Software-Entwicklung im Phasenmodell ähneln. In der Regel sind das die folgenden vier Phasen:

1. *Definitionsphase*[8],

[8] Auch: Vorbereitungsphase (oose.de GmbH, 1999), Konzeptphase (Balzert, 2000a, Seite 223)

2. *Entwurfsphase*[9],
3. *Konstruktionsphase* und
4. *Einführungsphase*.

Sie gliedern das Software-Projekt in mehrere zeitlich aufeinander folgende beziehungsweise sachlogisch zusammengehörende Abschnitte und stellen die oberste Planungseinheit dar. Die in jeder Phase zu erzielenden Ergebnisse sowie die damit verbundenen Aufwände, Kosten, Termine und Ressourcen sind in dem Projektplan festgehalten. Jede Phase besteht aus mindestens einer Iteration. Die Definitionsphase wird in der Regel wegen ihrer verhältnismäßig kurzen Laufzeit nicht in mehrere Iterationen unterteilt. Für die Entwurfsphase sind normalerweise ein bis zwei und für die Konstruktionsphase vier bis neun Iterationen vorgesehen. Die Einführungsphase besteht normalerweise aus einer Iteration. Sie kann aber auch aus mehreren Iterationen bestehen, zum Beispiel wenn das neue Software-System akkumulativ in das Unternehmen eingeführt werden soll. Die Anzahl der Iterationen in jeder Phase ist abhängig von dem Umfang des Projektes und dem jeweiligen Aufwand, der für die Erstellung des Software-Systems in dieser Phase zu betreiben ist.

3.4.1.1. Unified Software Development Process

Der in (Jacobson u. a., 1999) beschriebene Unified Software Development Process stellt den allgemeinsten Ansatz der auf Workflows basierenden Vorgehensmodelle dar. Der Software-Entwicklungsprozess ist anwendungsfallgetrieben, architekturzentriert und wird iterativ und inkrementell geführt. Der Unified Software Development Process besteht aus den genannten vier Phasen und wird durch die fünf Workflows *Anforderungsbestimmung*, *Analyse*, *Design*, *Implementierung* und *Test* definiert. Im Wesentlichen stellt der Unified Software Development Process ein Rahmengerüst zur Verfügung, mit dem komplexe und nur schwer überschaubare Software-Systeme handhabbar gemacht werden können. Des Weiteren besteht die Möglichkeit, den Unified Software Development Process an die eigenen Bedürfnisse anzupassen und zu konfigurieren. So können die verschiedenen Workflows für ein Projekt verkürzt werden oder es werden neue, speziell auf den Anwendungsfall zugeschnittene Workflows hinzugefügt. Workflows für das Software-Management und die Software-Qualitätssicherung sind im Unified Software Development Process nicht enthalten.

3.4.1.2. Rational Unified Process

Die Firma *Rational Software*[10] hat mit dem Rational Unified Process ein Vorgehensmodell für die Erstellung betrieblicher Software-Systeme entwickelt (siehe Kruchten, 2000). Das Vorgehensmodell ist eine spezielle und detaillierte Instanz des Unified Software Development Process und wird von Rational Software als eigenständiges Produkt vertrieben und

[9] Auch: Spezifikationsphase (Balzert, 2000a, Seite 223)
[10] Rational Software Corporation (siehe: http://www.rational.com/)

weiterentwickelt. Im Unterschied zum Unified Software Development Process werden die Rollen im Rational Unified Process mit *Worker* bezeichnet. Des Weiteren basiert die Software-Entwicklung nach dem Rational Unified Process auf objektorientierten Konzepten, Prinzipien und Techniken.

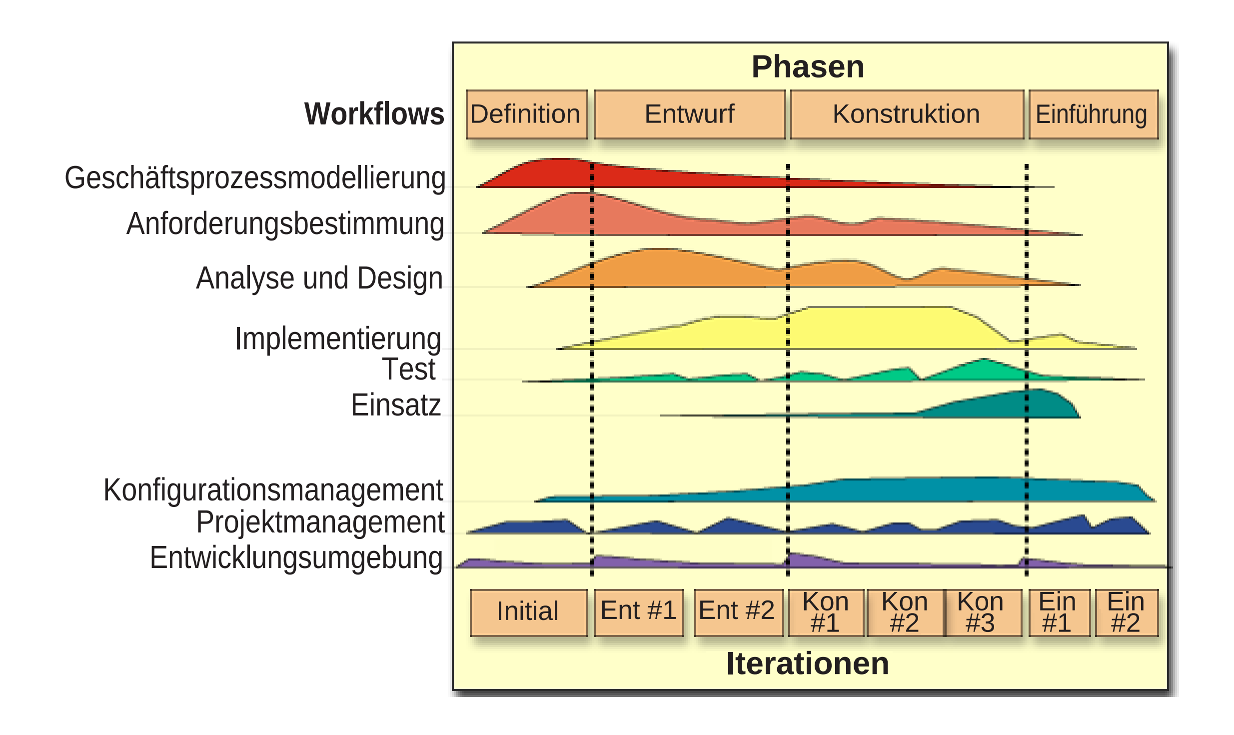


Abbildung 3.4.: Die Phasen und Workflows des Rational Unified Process

Wie aus Abbildung 3.4 zu ersehen ist, wird die zweidimensionale Struktur des Rational Unified Process durch die Phasen und den dazu orthogonal stehenden Workflows aufgespannt. Dabei stellen die Phasen beziehungsweise Iterationen eine zeitliche Einteilung und die Workflows eine inhaltliche Organisation des Software-Entwicklungsprozesses dar. Jede Phase besteht aus mindestens einer Iteration und setzt sich aus den neun Workflows zusammen. Die Workflows lassen sich des Weiteren in zwei logische Gruppen aufteilen. Die erste Gruppe von Workflows wird als *Engineering Workflows* bezeichnet und besteht aus dem Workflow zur *Geschäftsprozessmodellierung*, *Anforderungsbestimmung*, *Analyse und Design*, *Implementierung*, *Test* und *Einsatz*[11]. Die zweite Gruppe von Workflows besteht aus dem Workflow zum *Konfigurationsmanagement*, *Projektmanagement* und der *Entwicklungsumgebung* und werden *Supporting Workflows* genannt. Des Weiteren ist aus Abbildung 3.4 zu ersehen, welche Aufwandsanteile die Workflows des Rational Unified Process in den einzelnen Iterationen, das heißt über den Verlauf der Software-Entwicklung haben.

[11] Auch: Einsatz und Verteilung (oose.de GmbH, 1999)

3.4.1.3. Object Engineering Process

Der Object Engineering Process wurde von der Firma *oose.de*[12] entwickelt und stellt einen praxisorientierten und praxiserprobten Vorgehensleitfaden für die objektorientierte Software-Entwicklung dar (oose.de GmbH, 1999). Aufbau und Struktur dieses Vorgehensmodells folgen dem Unified Software Development Process. Die Inhalte des Vorgehensmodells, also die Workflows, Aktivitäten und Artefakte, sind jedoch unabhängig von dem Unified Software Development Process festgelegt. Der Object Engineering Process ist auf die Entwicklung individueller betrieblicher Software-Systeme ausgerichtet und kann an die unternehmensspezifischen Bedürfnisse angepasst werden. Da das Vorgehensmodell eine bestimmte Ausprägung der Anwendungsarchitekur unterstellt, wird der Object Engineering Process nicht mehr als architekturzentriert, sondern als *architekturspezifisch* bezeichnet.

Des Weiteren existieren Ergänzungen beziehungsweise Erweiterungen zum Object Engineering Process, wie zum Beispiel das *Timepacing–Verfahren* (siehe Hruschka u. a., 2001, Seite 134 f.). Dieses stellt im Wesentlichen eine Kombination bereits bekannter und erprobter Techniken und Verfahren des Projektmanagements dar (siehe Oestereich, 1999). Eine Kernaussage des Timepacing–Verfahrens ist, dass iteratives Vorgehen bei der Software-Entwicklung nur in Verbindung mit dem sogenannten *Timeboxing* funktionieren kann. Beim Timeboxing hat der Endtermin einer Iteration eine höhere Priorität als das gelieferte Ergebnis. Im Software-Entwicklungsprozess bedeutet dies konkret, dass zum vereinbarten Termin geliefert wird, egal was (Hruschka u. a., 2001, Seite 131). Ebenfalls wichtiger Bestandteil des Timepacing–Verfahrens ist die richtige Wahl der Zeitdauer der Iterationen. Ist die Zeitdauer einer Iteration zu kurz gewählt, so wird die Iteration unproduktiv. Die Zeitdauer darf aber auch nicht zu lang sein, da sonst die gesteckten Ziele und vereinbarten Aufgaben zu abstrakt werden (siehe Hruschka u. a., 2001).

3.4.2. Extreme Programming

Das in (Beck, 2000; Reißing, 2000) beschriebene Vorgehensmodell des Extreme Programming (XP) besteht aus einer Reihe von *Praktiken* zur Software-Entwicklung, die nur in ihrer Gesamtheit sinnvoll sind. Der Software-Entwicklungsprozess gliedert sich beim Extreme Programming in Iterationen von ein bis drei Wochen. Mehrere Iterationen ergeben ein *Release*. Ziel des ersten Release ist die Erstellung eines funktionierenden Pilotsystems, dass die wichtigsten Funktionalitäten beinhaltet. Anschließend wird das Pilotsystem inkrementell weiterentwickelt. Nach jedem Release wird eine Evaluation durch den Endanwender vorgenommen. Die Planung der einzelnen Iterationen erfolgt beim Extreme Programming in der Praktik des *Planungsspiels*. Im Planungsspiel werden die Anforderungen an das Software-System in rudimentären Anwendungsfällen (*storys*) niedergeschrieben. Des Weiteren werden Prioritäten bezüglich der Anforderungen festgelegt. Eine zentrale Praktik des Extreme Programming sind die *automatisierten Tests*. Ohne *automatisierte Tests* können die kurzen Iterationszyklen des Extreme Programming nicht eingehalten werden oder die Iterationen

[12] oose.de Dienstleistungen für innovative Informatik GmbH (siehe: http://www.oose.de/gmbh/)

werden unproduktiv. Im *Entwurf* sind immer die einfachsten Lösungen zu suchen, die nur die Anforderungen der jeweiligen Iteration abdecken. Durch *Refaktorisierung* (*refactoring*) wird der Entwurf des bestehenden Software-Systems unter Beibehaltung der Semantik vereinfacht. Aufgrund des einfachen Entwurfs sieht das Extreme Programming keine echte System-Spezifikation vor. Um den einfachen Entwurf auszugleichen wird aber gefordert, über den gesamten Software-Entwicklungsprozess einen *Kundenvertreter im Team* verfügbar zu haben. Weitere zentrale Praktik ist das *Programmieren in Paaren* (*pair-programming*), die in den Phasen Entwurf, Implementierung und Test verwendet wird. Während einer von zwei Entwicklern programmiert, prüft der andere den Quelltext auf Fehler und kontrolliert den Entwurf auf Konsistenz. Eine weitere Praktik des Extreme Programming ist die *kontinuierliche Code-Integration*. Neu entwickelter oder geänderter Quelltext wird alle paar Stunden in das aktuelle System integriert. Auf diese Weise ist immer ein lauffähiges Software-System verfügbar. Ergänzt wird diese Praktik durch die Idee des *gemeinsamen Code-Eigentums*. Jedes Entwicklerpaar darf jederzeit und überall im Quelltext Änderungen vornehmen. Zur Erleichterung der Arbeit in Paaren und des gemeinsamen Code-Eigentums, sieht das Vorgehensmodell verbindliche *Programmierrichtlinien* vor.

In der Literatur existieren Erweiterungen beziehungsweise Ergänzungen zum Extreme Programming. In (Barchfeld u. a., 2000) wird untersucht, ob das Extreme Programming als eine Instanz des Rational Unified Process angesehen werden kann, da es die wichtigsten Prinzipien des Rational Unified Process erfüllt. Des Weiteren wird betrachtet, unter welchen Rahmenbedingungen es sinnvoll erscheint das Extreme Programming um Elemente des Rational Unified Process zu erweitern und umgekehrt.

Die Praktiken des Extreme Programming sind nicht ohne Auswirkung auf das Projektmanagement. In (Oestereich, 1999) wird daher auch von *Extreme Projectmanagement* (XPM) gesprochen. Dabei handelt es sich um die Integration von Praktiken des Extreme Programming in das dort vorgestellte Timepacing–Verfahren (siehe dazu auch die Erweiterungen zum Object Engineering Process in Abschnitt 3.4.1.3).

3.4.3. Bewertung der modernen Vorgehensmodelle

Die im Unified Software Development Process, Rational Unified Process und Object Engineering Process vorgenommene Aufteilung des Software-Entwicklungsprozesses in inhaltlich voneinander getrennte Workflows kommt der hohen Anzahl der zu berücksichtigenden Aspekte bei der Entwicklung virtueller Labore sehr entgegen. Des Weiteren wird durch die Zuordnung der Rollen, Aktivitäten und Artefakte die Arbeitsverteilung stark vereinfacht. Darüber hinaus ermöglichen die Workflows eine übersichtliche und leicht verständliche Darstellung der durchzuführenden Aktivitäten.

Gegenüber dem Unified Software Development Process enthält der Rational Unified Process noch weitere Workflows für das Konfigurationsmanagement, Projektmanagement und die Entwicklungsumgebung. Damit wird der Nachteil des Unified Software Development Process, nicht alle drei Bereiche der klassischen Software-Technik abzudecken, mit dem Rational Unified Process behoben. Der Rational Unified Process enthält zudem noch einen

weiteren Workflow für die Geschäftsprozessmodellierung, mit dem das Vorgehensmodell auf die Entwicklung betrieblicher Software-Systeme spezialisiert wird. Diese Spezialisierung stellt gegenüber dem Unified Software Development Process keinen Nachteil dar, da der Workflow zur Geschäftsprozessmodellierung ohne weiteres aus dem Vorgehensmodell entfernt werden kann, falls dieser nicht benötigt wird. Der Object Engineering Process ist dem Rational Unified Process sehr ähnlich. Wie der Rational Unified Process, so ist auch der Object Engineering Process auf die Entwicklung betrieblicher Software-Systeme spezialisiert und wird ebenfalls als eigenständiges Produkt vermarktet und weiterentwickelt. Während der Object Engineering Process allerdings ein reines Vorgehensmodell darstellt, sieht der Rational Unified Process einen ganzheitlichen Ansatz vor. Neben der Zuordnung von Rollen, Aktivitäten und Artefakte findet im Rational Unified Process auch eine Zuordnung der einzusetzenden Entwicklungswerkzeuge (*CASE-Werkzeuge*) statt, ohne das der Entwickler auf die Verwendung dieser Werkzeuge angewiesen ist.

Wesentlicher Vorteil der auf Workflows basierenden modernen Vorgehensmodellen gegenüber den klassischen und speziellen Vorgehensmodellen ist, dass diese sich relativ einfach an die Bedürfnisse einer konkreten Anwendung erweitern und anpassen lassen. Außerdem ist in den auf Workflows basierenden modernen Vorgehensmodellen bereits ein Rollenkonzept integriert. Daher eignen sich diese Vorgehensmodelle hervorragend für die Entwicklung virtueller Labore.

Das Vorgehensmodell des Extreme Programming kann nicht für die Entwicklung virtueller Labore eingesetzt werden, da eine der wesentlichen Praktiken des Extreme Programming bei virtuellen Laboren nicht anwendbar ist. So lassen sich die im Extreme Programming geforderten automatisierten Tests nur für den Simulator erstellen. Die aufwendigen Tests des virtuellen Labors, das heißt das Testen des zeitlichen Zusammenspiels der multimedialen Komponenten, lassen sich wegen der hohen Interaktivität des Software-Systems kaum vereinfachen oder automatisieren. Daher können die kurzen Iterationszyklen des Extreme Programming nicht eingehalten werden. Somit ist das Vorgehensmodell des Extreme Programming als solches für die Entwicklung virtueller Labore ungeeignet. Dennoch macht es durchaus Sinn, bestimmte Praktiken in das Vorgehensmodell für virtuelle Labore zu übernehmen, wie das Programmieren in Paaren für das Simulationsmodell, die Refaktorisierung der Architektur und für den Simulator die automatisierten Tests. Es soll außerdem nicht unerwähnt bleiben, dass durch den Wegfall einer ausgiebigen Analyse- und Definitionsphase, das Vorgehensmodell des Extreme Programming im harten Widerspruch zur gängigen Lehrmeinung der Software-Technik steht (siehe Hruschka u. a., 2001).

3.5. Lösungsansatz für virtuelle Labore: der VirtLab–Prozess

Eine Übersicht der Bewertung der klassischen, speziellen und der modernen Vorgehensmodelle nach dem in Kapitel 3.1 vorgestellten Bewertungsschema für Vorgehensmodelle im Bezug auf den Anwendungsfall virtuelle Labore ist in den Tabellen 3.2, 3.3 und 3.4 dargestellt.

Bewertungskriterium	**Phasenmodell**	**Iteriertes Phasenmodell**	**Prototypenmodell**	**Evolutionäre Software-Entwicklung**	**Transformationelle Software-Entwicklung**
Heterogenität des Teams	n.v.	n.v.	n.v.	n.v.	n.v.
Einbez. des Auftraggebers	-	(-)	+	+	o
Einbez. der Endanwender	-	(-)	+	+	o
Benutzungsoberfläche	o	o	+	+	(-)
Medienproduktion	n.v.	n.v.	n.v.	n.v.	n.v.
Erstellung der Didaktik	n.v.	n.v.	n.v.	n.v.	n.v.
Adaptierbarkeit	(-)	(-)	(-)	(-)	(-)
Wiederverw. best. Entw.	(-)	o	o	o	o
Angem. der Tests	(-)	o	o	o	+
Angem. der Risikoanalysen	(-)	o	+	+	-
Bewertungsurteil	**-**	**(-)**	**(-)**	**(-)**	**(-)**

Bewertungskriterium	**Spiralmodell**	**V–Modell**	**Objektorientierte Software-Entwicklung**	**Catalysis**
Heterogenität des Teams	o	+	n.v.	n.v.
Einbez. des Auftraggebers	o	o	o	o
Einbez. der Endanwender	o	+	o	o
Benutzungsoberfläche	o	o	o	o
Medienproduktion	n.v.	n.v.	n.v.	n.v.
Erstellung der Didaktik	n.v.	n.v.	n.v.	n.v.
Adaptierbarkeit	(-)	(-)	o	o
Wiederverw. best. Entw.	o	o	++	++
Angem. der Tests	(-)	o	o	o
Angem. der Risikoanalysen	o	(-)	o	o
Bewertungsurteil	**(-)**	**(-)**	**(-)**	**(-)**

Tabelle 3.2.: Bewertung der klassischen Vorgehensmodelle

Insgesamt lässt sich feststellen, dass die Vorgehensmodelle der klassischen Software-Technik, die speziellen Vorgehensmodelle und die Vorgehensmodelle der modernen Software-Technik entweder zu abstrakt und zu allgemein sind oder nur einen Teil der in Kapitel 3.1 genannten Bewertungskriterien berücksichtigen, als dass sie nutzbringend für die Entwicklung virtueller Labore eingesetzt werden könnten. Aus den in Kapitel 3.2 bis 3.4 betrachteten

Bewertungskriterium	**Vorgehensmodell für MMS (Sawhney)**	**Vorgehensmodell zur Entwicklung von LLS (Yass)**	**Vorgehensmodell für die Entwicklung von MMLLS (Nagl u.a.)**
Heterogenität des Teams	n.v.	n.v.	(-)
Einbez. des Auftraggebers	o	o	o
Einbez. der Endanwender	+	(-)	o
Benutzungsoberfläche	+	+	+
Medienproduktion	+	n.v.	o
Erstellung der Didaktik	o	+	+
Adaptierbarkeit	o	o	o
Wiederverw. best. Entw.	o	o	o
Angem. der Tests	o	o	o
Angem. der Risikoanalysen	o	o	o
Bewertungsurteil	**o**	**(-)**	**o**

Tabelle 3.3.: Bewertung der speziellen Vorgehensmodelle

Bewertungskriterium	**Unified Software Development Process**	**Rational Unified Process**	**Object Engineering Process**	**Extreme Programming**
Heterogenität des Teams	+	+	+	o
Einbez. des Auftraggebers	+	+	+	++
Einbez. der Endanwender	o	+	+	++
Benutzungsoberfläche	o	o	o	o
Medienproduktion	n.v.	n.v.	n.v.	n.v.
Erstellung der Didaktik	n.v.	n.v.	n.v.	n.v.
Adaptierbarkeit	+	++	+	(-)
Wiederverw. best. Entw.	+	+	o	(-)
Angem. der Tests	o	o	o	n.v.
Angem. der Risikoanalysen	+	+	+	(-)
Bewertungsurteil	**o**	**o**	**o**	**(-)**

Tabelle 3.4.: Bewertung der modernen Vorgehensmodelle

Vorgehensmodellen lassen sich zwei prinzipielle Lösungsansätze eines Vorgehensmodells für virtuelle Labore ableiten. Das sind konkret die folgenden beiden Ansätze:

1. Einbettung der objektorientierten Software-Entwicklung oder Anwendung von Catalysis in eines der in Kapitel 3.3 vorgestellten speziellen Vorgehensmodelle.

2. Adaption und Erweiterung eines auf Workflows basierenden modernen Vorgehensmodells aus Kapitel 3.4.

Wesentlicher Vorteil des ersten Ansatzes gegenüber dem zweiten Ansatz ist, dass bereits die multimedialen und didaktischen Aspekte berücksichtigt werden. Die speziellen Vorgehensmodelle sind aber eher auf die Entwicklung von hypermedialen Multimedia-Systemen beziehungsweise klassischen Lehr- und Lernsystemen zugeschnitten. Eine Anpassung eines speziellen Vorgehensmodells für die Entwicklung von virtuellen Laboren kommt daher bis auf die Bestandteile der klassischen Software-Technik einer Neuentwicklung des Vorgehensmodells gleich. Der zweite Ansatz der auf Workflows basierenden Vorgehensmodelle kann dem gegenüber wesentlich besser auf den Anwendungsfall virtuelle Labore angepasst werden. Die Workflows dieser modernen Vorgehensmodelle sind um die Aspekte virtueller Labore anzupassen und um die Medienproduktion und die Erstellung der Didaktik zu erweitern. Des Weiteren wird von den auf Workflows basierenden Vorgehensmodellen die Heterogenität des Entwicklerteams bereits unterstützt. Die Argumente bezüglich der Anpassbarkeit des Vorgehensmodells und der Heterogenität des Entwicklerteams sind letztendlich für die Entscheidung ausschlaggebend, dass der zweite Ansatz weiterverfolgt wird.

Als Grundlage des Vorgehensmodells für virtuelle Labore bietet sich zunächst der Unified Software Development Process an. Allerdings werden nicht alle Bereiche der Software-Technik mit den Workflows des Unified Software Development Process berücksichtigt. Aus diesem Grund beinhaltet der Rational Unified Process weitere Workflows für das Konfigurationsmanagement, Projektmanagement und die Entwicklungsumgebung sowie einen Workflow für die Geschäftsprozessmodellierung. Für die Entwicklung von virtuellen Laboren wird der Workflow zur Modellierung von Geschäftsprozessen nur zum Teil benötigt. So können die für die Entwicklung von betrieblichen Software-Systemen besonders wichtigen Aktivitäten zur Automatisierung von Geschäftsprozessen sowie zur Zuordnung von Verantwortlichkeiten und Rollen für den Anwendungsfall virtuelle Labore entfernt werden. Die restlichen Aktivitäten des Workflows zur Geschäftsprozessmodellierung werden nicht entfernt, da sich mit diesen die im virtuellen Labor durchzuführenden Versuche mit Hilfe von Geschäftsprozessen beschreiben lassen (vergleiche Anhang A). Da von dem Rational Unified Process die didaktischen Aspekte virtueller Labore völlig unberücksichtigt bleiben, wird dieser um einen Workflow des *Tutorkonzeptes* erweitert. Dieser umfasst die Entwicklung der inhaltlichen und didaktischen Bestandteile des intelligenten Tutorsystems und der Versuche des virtuellen Labors. Sämtliche für virtuelle Labore relevanten Aktivitäten der Geschäftsprozessmodellierung werden in den Workflow des Tutorkonzeptes integriert, so dass der Workflow zur Geschäftsprozessmodellierung vollständig aus dem Rational Unified Process entfernt werden kann. Außerdem wird der Rational Unified Process noch um

den Workflow der *Medienproduktion* ergänzt. Dieser beinhaltet sämtliche Arbeitsschritte, die bei der Planung und Herstellung der Medien durchzuführen sind. Die softwaretechnischen Anteile des intelligenten Tutorsystems und des Simulators werden bereits durch die anderen Workflows des Rational Unified Process abgedeckt. Aus diesem Grund wird dem Vorgehensmodell kein weiterer Workflow für die Erstellung des Simulators hinzugefügt.

Insgesamt besteht das Vorgehensmodell für virtuelle Labore also aus dem Rational Unified Process (ohne die Geschäftsprozessmodellierung), dem Workflow für das Tutorkonzept und dem Workflow für die Medienproduktion. Damit wird insgesamt eine gleich hohe Gewichtung auf die Aspekte Medienproduktion, Tutorsystem und Simulator sichergestellt. Die Berücksichtigung der Interdisziplinarität des Entwicklerteams erfolgt durch die Zuordnung von Rollen zu den durchzuführenden Aktivitäten und Artefakten. Dieser Lösungsansatz für ein Vorgehensmodell und die zugehörende Entwicklungsmethodik für virtuelle Labore wird im Folgenden als der *VirtLab*–Prozess bezeichnet.

Der Simulator eines virtuellen Labors stellt sehr hohe Ansprüche an die Interaktivitätsmöglichkeiten der Benutzungsoberfläche. Der Einsatz eines modernen Autorensystems mit integrierter Programmiersprache ist daher für die Entwicklung virtueller Labore unabdingbar. Dadurch wird allerdings die Entwicklung des virtuellen Labors unter Umständen auf ein Produkt beziehungsweise eine Produktpalette eines Herstellers gebunden, sofern keine Austauschmöglichkeiten zu Autorensystemen anderer Hersteller bestehen. In der Regel reduziert sich durch die Verwendung eines Autorensystems der Entwicklungsaufwand im erheblichen Maße, da aktuelle Autorensysteme mit einer integrierten Skriptsprache[13] speziell auf die Erstellung von multimedialen Systemen zugeschnitten sind und die Skriptsprache auch hohen Ansprüchen bezüglich Interaktivität gerecht wird. Autorensysteme unterstützen allerdings nicht den gesamten Entwicklungsprozess multimedialer Anwendungen, sondern treten lediglich während der Implementierungsphase und in Ausschnitten während der Entwurfsphase und der Testphase in Erscheinung (siehe Boles und Schlattmann, 1998). Das Konfigurationsmanagement ist im Allgemeinen nicht in Autorensysteme integriert und kooperative Arbeitsformen werden ebenfalls nicht unterstützt. Daher erscheint es für die Entwicklung virtueller Labore sinnvoll, auf weitere Werkzeuge zurückzugreifen. Wie bereits in Kapitel 3.4.3 der Bewertung der modernen Vorgehensmodelle angedeutet, macht es durchaus Sinn bestimmte Praktiken des Extreme Programming in den *VirtLab*–Prozess zu übernehmen. Für die Erstellung des Simulators bietet sich die Praktik der automatisierten Tests an. Das ist gerade dann der Fall, wenn der Simulator mit Hilfe des transformationellen Vorgehensmodells konzipiert und implementiert werden soll. Zur Optimierung des Frameworks des virtuellen Labors kann die Praktik der Refaktorisierung angewendet werden. Das Programmieren in Paaren ist bei einem besonders komplexen Simulationsmodell zu empfehlen, zum Beispiel wenn der Simulator auf einer sehr abstrakten oder schwer verständlichen Theorie basiert. Der im Extreme Programming geforderte ständige Kundenvertreter im Entwicklerteam kann bei virtuellen Laboren beispielsweise durch einen Fachexperten vertreten sein.

---

[13]Zum Beispiel die Skriptsprache *Lingo* des Autorensystems *Director* von *Macromedia* (siehe: http://www.macromedia.com/software/director/)

4. Die Phasen und Meilensteine des VirtLab–Prozess

In diesem und den folgenden Kapiteln wird der in Kapitel 3.5 vorgeschlagene Lösungsansatz eines Vorgehensmodells und der zugehörenden Entwicklungsmethodik für virtuelle Labore (kurz: *VirtLab*–Prozess) im Detail beschrieben. Dazu werden in diesem Kapitel zunächst die Phasen und Meilensteine des *VirtLab*–Prozess eingeführt. Anschließend werden in Kapitel 5 die am Software-Entwicklungsprozess beteiligten Rollen beschrieben. Danach wird in Kapitel 6 das Metaobjektmodell für virtuelle Labore als wichtigster Grundbaustein der Entwicklungsmethodik vorgestellt. In Kapitel 7 werden dann die Workflows des Vorgehensmodells zusammen mit den zugehörenden Aktivitäten und Artefakten der Entwicklungsmethodik beschrieben. Des Weiteren wird folgende Notation festgelegt: Die am Software-Entwicklungsprozess beteiligten Rollen werden mit dem Symbol Rolle gekennzeichnet. Die Workflows des Vorgehensmodells erhalten die Beschriftung Workflow. Des Weiteren werden die Iterationsmeilensteine (IMS) mit IMS und die Phasenmeilensteine (PMS) mit PMS gekennzeichnet. Darüber hinaus werden die Aktivitäten mit Aktivität und die Artefakte mit Artefakt beschriftet.

Der *VirtLab*–Prozess besteht nach Abbildung 4.1 aus den vier nacheinander ablaufenden Phasen *Definition*, *Entwurf*, *Konstruktion* und *Einführung* (vergleiche Kruchten, 2000, Seite 62). Eine Phase fasst eine Menge von Aktivitäten und Artefakte zu einer logischen Einheit zusammen. Am Ende jeder Phase steht mindestens ein Phasenmeilenstein, der die in der Phase zu erzielenden Artefakte definiert (siehe oose.de GmbH, 1999). Jede Phase setzt sich aus mindestens einer Iteration zusammen. Am Ende einer Iteration steht ein Iterationsmeilenstein. Dieser definiert wichtige Zwischenergebnisse in einer Phase. Ergeben sich nach der Einführungsphase seitens des Kunden Änderungswünsche, neue Anforderungen oder sind Fehler zu beseitigen, so tritt die Entwicklung des virtuellen Labors in die Phase der Evolution[1] ein. Das bedeutet, dass wieder bei der Definitionsphase begonnen wird. Als Zeitrahmen für ein beispielsweise zwei Jahre andauerndes Projekt schlägt (Kruchten, 2000, Seite 63 f.) eine Definitionsphase von acht bis zehn Wochen, eine Entwurfsphase von sieben Monaten, eine Konstruktionsphase von zwölf Monaten und eine Einführungsphase von wiederum acht bis zehn Wochen vor. In den nachfolgenden Kapiteln 4.1 bis 4.4 werden die einzelnen Phasen und Meilensteine des *VirtLab*–Prozess kurz beschrieben (nach Kruchten, 2000; oose.de GmbH, 1999).

[1] Auch: Betriebsphase (siehe oose.de GmbH, 1999), Produktionsphase (siehe Ambler, 2001, Seite 20)

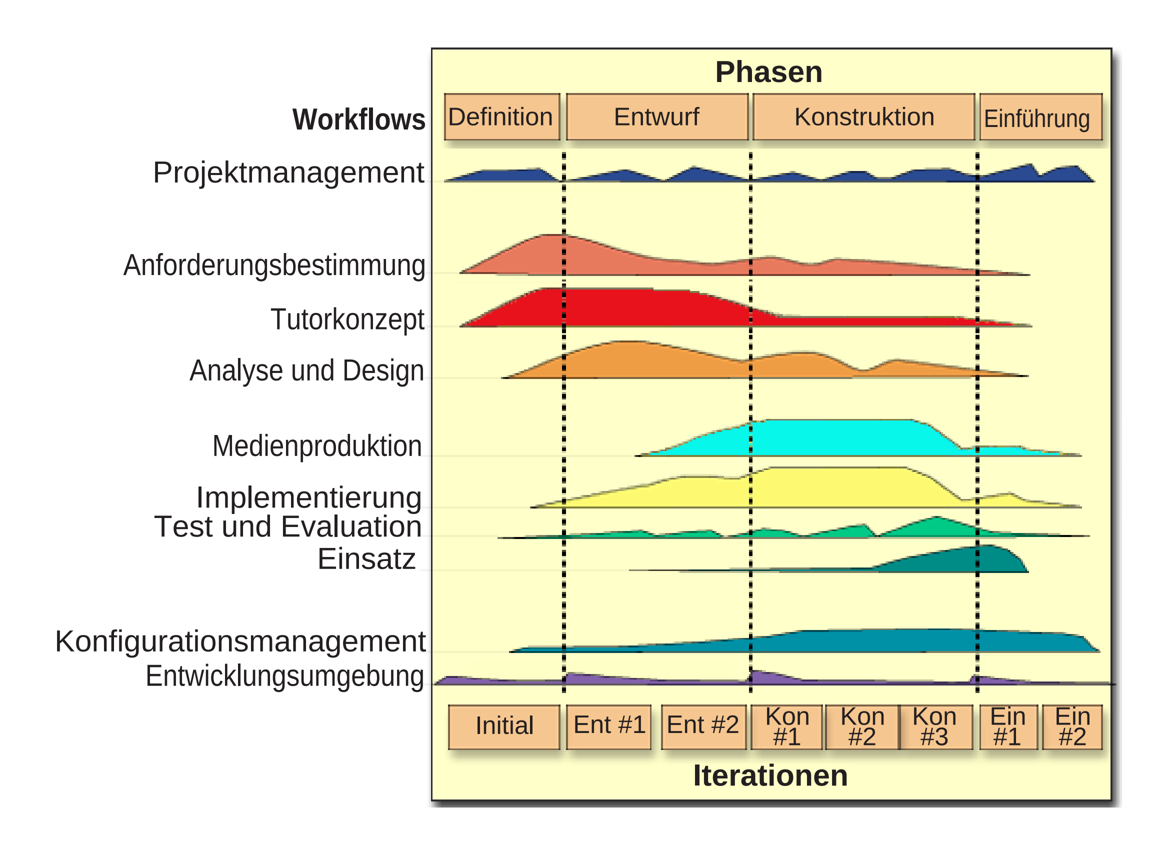


Abbildung 4.1.: Die Phasen und Workflows des Vorgehensmodells für virtuelle Labore

4.1. Definitionsphase

In der Definitionsphase werden die grundlegenden Anforderungen an das virtuelle Labor erhoben und im *Pflichtenheft*[Artefakt] festgehalten. Darüber hinaus ist der Zeit- und Kostenrahmen zu klären und im *Gesamtentwicklungsplan*[Artefakt] festzuhalten. Außerdem ist eine Entscheidung über die Durchführung der Entwicklung herbeizuführen. Es ist nicht Ziel der Definitionsphase die zu modellierende Domäne bereits in detaillierter Form zu beschreiben. Es sind aber alle Anwendungsfälle (*use-cases*) des virtuellen Labors zu identifizieren und im *Anwendungsfall-Modell*[Artefakt] (*use-case model*) festzuhalten. Die wichtigsten Anwendungsfälle sind die zur Auswahl und Durchführung der einzelnen Versuche. Die Spezifikation dieser Anwendungsfälle ist dabei durch die entsprechenden Versuchsprotokolle bereits gegeben. Wegen der Interdisziplinarität des Entwicklerteams ist ein gemeinsames Verständnis von der Domäne des virtuellen Labors von großer Bedeutung und eine wesentliche Voraussetzung für den Erfolg des Projekts. Dazu sind mögliche Mehrdeutigkeiten der zu modellierenden Domäne zu erkennen und aus dem Weg zu räumen. Die Ergebnisse dieser Aktivität werden im *Gesamtglossar*[Artefakt] festgehalten.

Die Definitionsphase wird immer dann durchgeführt, wenn ein neues virtuelles Labor erstellt (siehe Balzert, 2000b, Seite 58 bis 60) oder umfangreiche Wartungsarbeiten an

einem existierenden virtuellen Labor vorgenommen werden. Bestehen Zweifel an der Durchführbarkeit von Anforderungen, so werden *Vorstudien*[Artefakt][2] erstellt oder einfache Prototypen[3] entwickelt, um eine Entscheidung bezüglich der Realisierbarkeit herbeizuführen. Das wichtigste Ergebnis der Definitionsphase ist die Entscheidung, ob das virtuelle Labor entwickelt werden soll oder nicht.

Daher sind die Phasenmeilensteine der Definitionsphase:

- *Vereinbarung über den Projektauftrag*[PMS] und
- *Vorstudie verfügbar*[PMS] (optional).

Zur Erreichung der Phasenmeilensteine müssen die folgenden Artefakte vorliegen:

- *Pflichtenheft*[Artefakt],
- *Vorstudie*[Artefakt] (optional) sowie
- *Gesamtentwicklungsplan*[Artefakt] (in einer initialen Version),
- *Gesamtglossar*[Artefakt] (in einer ersten Version) und
- *Anwendungsfall-Modell*[Artefakt] (alle Anwendungsfälle wurden erfasst).

4.2. Entwurfsphase

In der Entwurfsphase wird eine detaillierte Analyse des zu entwickelnden virtuellen Labors durchgeführt. Dazu sind die funktionalen und nicht-funktionalen Anforderungen, sofern nicht schon in der Definitionsphase geschehen, detailliert zu spezifizieren. Außerdem werden die grundlegende *Architektur*[Artefakt] in Form eines Frameworks und das *didaktische Konzept*[Artefakt] des virtuellen Labors festgelegt sowie ein *Gesamtentwicklungsplan*[Artefakt] erstellt. Das wichtigste Ergebnis der Entwurfsphase ist ein nahezu[4] vollständig spezifiziertes *Anwendungsfall-Modell*[Artefakt], das heißt es liegen wenigstens für jeden Versuch ein entsprechendes *annotiertes Versuchsprotokoll*[Artefakt] und ein *Drehbuch*[Artefakt] vor. In (oose.de GmbH, 1999) wird vorgeschlagen, am Ende der Entwurfsphase die Ergebnisse dem *Auftraggeber*[Rolle] vorzustellen (externes *Review*). Da die Entwicklung eines virtuellen Labors ein großes finanzielles Risiko darstellt, erscheint es sinnvoll, zu diesem Zeitpunkt über die weitere Durchführung oder den Abbruch des Projekts zu entscheiden.

Umfasst die Entwurfsphase mehr als eine Iteration, so wird ein Iterationsmeilenstein eingefügt. An diesem Meilenstein werden wichtige Zwischenergebnisse der Entwurfsphase

[2] Auch: Durchführbarkeitsstudie (Balzert, 2000b, Seite 69)

[3] Das können Prototypen zur Demonstration der Benutzungsoberfläche sein (*Oberflächenprototypen*), aber auch Prototypen, bei denen bereits ein Teil der Funktionalität des virtuellen Labors realisiert wurden.

[4] In (Balzert, 2000a, Seite 224) und (Larman u. a., 2001, Seite 33) wird gefordert, dass ungefähr 80% der Anwendungsfälle komplett spezifiziert sind.

gebunden, wie zum Beispiel die Spezifikation zentraler Anwendungsfälle. Nach Durchführung einer Iteration wird ein *Ergebnisbericht der Iteration*[Artefakt] erstellt. Des Weiteren ist der *Gesamtentwicklungsplan*[Artefakt] zu aktualisieren.

Der Iterationsmeilenstein der Entwurfsphase ist das *Iterationsende*[IMS]. Zur Erreichung des Iterationsmeilensteins müssen die folgenden Artefakte vorliegen:

- *Ergebnisbericht der Iteration*[Artefakt] und
- *Gesamtentwicklungsplan*[Artefakt] (aktualisiert).

Die Phasenmeilensteine in der Entwurfsphase sind:

- *Architektur festgelegt*[PMS] und
- *Didaktik festgelegt*[PMS].

Zur Erreichung der Phasenmeilensteine müssen die folgenden Artefakte vorliegen:

- *Architektur*[Artefakt],
- *didaktisches Konzept*[Artefakt] und
- *Anwendungsfall-Modell*[Artefakt] (zu ungefähr 80% spezifiziert).

4.3. Konstruktionsphase

In der Konstruktionsphase werden die Anforderungen und die *Architektur*[Artefakt] des virtuellen Labors erweitert und verfeinert. Innerhalb einer Iteration werden die Anforderungen bezüglich Analyse, Design, Implementierung, Medienproduktion und Test nach Priorität geordnet und die entsprechenden Aktivitäten durchgeführt. Dabei nimmt die Gesamtfunktionalität des virtuellen Labors inkrementell von Iteration zu Iteration zu. Am Ende der Konstruktionsphase sind alle Anforderungen analysiert, entworfen, implementiert und getestet sowie alle Medien produziert und integriert worden, das heißt alle Versuche des virtuellen Labors sind umgesetzt. Die Konstruktionsphase endet mit einem ersten *externen Release*[Artefakt], also mit der Bereitstellung des virtuellen Labors für die Übernahme in die Einsatzumgebung (siehe Balzert, 2000a; oose.de GmbH, 1999).

In der Regel besteht die Konstruktionsphase aus mehreren Iterationen. Jeder Iterationsmeilenstein ist dabei mit einem neuen *internen Release*[Artefakt], den zugehörigen Testabläufen (α-*Tests*) und einer Demonstration des virtuellen Labors beim *Auftraggeber*[Rolle] verbunden. Nach Beendigung der Iteration wird ein *Ergebnisbericht der Iteration*[Artefakt] erstellt.

Der Iterationsmeilenstein der Konstruktionsphase ist das *interne Release*[IMS] (getestet). Zur Erreichung des Iterationsmeilensteins müssen die folgenden Artefakte vorliegen:

- *Ergebnisbericht der Iteration*[Artefakt] (inklusive *Test- und Evaluationsbericht*[Artefakt]),
- *Gesamtentwicklungsplan*[Artefakt] (aktualisiert) und
- *internes Release*[Artefakt] (inklusive des *Benutzungshandbuchs*[Artefakt] und der *technischen Dokumentation*[Artefakt], soweit dies möglich ist).

Der Phasenmeilenstein der Konstruktionsphase ist das *erste externe Release*[PMS], das heißt der Abschluss von Implementierung, Medienproduktion sowie internen Tests und Evaluation. Zur Erreichung des Phasenmeilensteins müssen die folgenden Artefakte vorliegen:

- *Ergebnisbericht der Iteration*[Artefakt] (inklusive *Test- und Evaluationsbericht*[Artefakt]),
- *Gesamtentwicklungsplan*[Artefakt] (aktualisiert) und
- erstes *externes Release*[Artefakt] (inklusive des vollständigen *Benutzungshandbuchs*[Artefakt] und der *technischen Dokumentation*[Artefakt]).

4.4. Einführungsphase

In der Einführungsphase wird das virtuelle Labor in Form eines *externen Release*[Artefakt] an den *Auftraggeber*[Rolle] und den *Endanwendern*[Rolle] ausgeliefert und dort unter realistischen Bedingungen getestet (*β-Tests*). Änderungswünsche bezüglich der Funktionalitäten und der Gestaltung der Benutzungsoberfläche sollten in der Einführungsphase nur noch in Detailfragen auftreten. Ist die Einführung des virtuellen Labors mit einem größeren organisatorischen Aufwand verbunden, zum Beispiel wenn das virtuelle Labor in Verbindung mit einem realen Laborpraktikum eingesetzt werden soll (vergleiche Kronberg, 1998, Seite 30), dann kann die Einführungsphase in mehreren Iterationen erfolgen. Um das virtuelle Labor in Verbindung mit einem realen Laborpraktikum einsetzen zu können, müssen die durchführenden Lehrenden beziehungsweise Betreuer des Laborpraktikums entsprechend geschult werden. Außerdem können empirische Untersuchungen bezüglich Akzeptanz und Lernerfolg der Praktikumsteilnehmer durchgeführt werden, um weitere Rückmeldungen und Anregungen zu erhalten (siehe dazu Eiwan, 1999; Kerres, 1998; Lottmann u. a., 2000). Eine Aufteilung der Einführungsphase in mehrere Iterationen bietet sich ebenfalls an, wenn es sich bei der Entwicklung des virtuellen Labors um eine Massenproduktion handelt. In diesem Fall empfiehlt es sich, eine besonders umfangreiche β-Testphase vor der eigentlichen Produktion durchzuführen, um die Ablauffähigkeit des virtuellen Labors auf den verschiedensten Computer-Systemen und Konfigurationen sicherzustellen. Wurde das virtuelle Labor in der Einsatzumgebung getestet und als betriebsfähig angesehen, so wird mit der Abnahme des letzten *externen Release*[Artefakt] durch den *Auftraggeber*[Rolle] die Einführungsphase abgeschlossen und damit das Projekt als beendet erklärt (siehe auch Frühauf u. a., 1991b, Seite 87).

Die Iterationsmeilensteine der Einführungsphase sind:

- *Abschluss der β-Tests*[IMS],
- *externes Release*[IMS] (beim *Auftraggeber*[Rolle] erfolgreich eingeführt) und
- *Abschluss der Schulung*[IMS] (optional).

Zur Erreichung des Iterationsmeilensteins müssen die folgenden Artefakte vorliegen:

- *Ergebnisbericht der Iteration*[Artefakt],
- *Gesamtentwicklungsplan*[Artefakt] (aktualisiert) und
- *externes Release*[Artefakt].

Der Phasenmeilenstein der Einführungsphase ist der *Projektabschluss*[PMS]. Zur Erreichung des Phasenmeilensteins muss das letzte *externe Release*[Artefakt] inklusive der endgültigen Version des *Benutzungshandbuchs*[Artefakt] und der *technischen Dokumentation*[Artefakt] vorliegen und vom *Auftraggeber*[Rolle] abgenommen worden sein.

5. Die Rollen des VirtLab–Prozess

Die an der Entwicklung von virtuellen Laboren beteiligten Personen nehmen bei der Durchführung der Aktivitäten die verschiedensten Rollen ein. Die allgemeinste Rolle, die eine Person des *Entwicklerteams* einnehmen kann, ist die des *Projektmitarbeiters*[Rolle][1]. Diese Rolle lässt sich in die vier Organisationseinheiten *Fachexperte*[Rolle], *Fachdidaktiker*[Rolle][2], *Medienspezialist*[Rolle] und *Informatiker*[Rolle] untergliedern. In den Kapiteln 5.1 bis 5.4 werden diese Organisationseinheiten näher beschrieben und gegebenenfalls weiter verfeinert. Als externe Rollen treten der *Auftraggeber*[Rolle] und die *Endanwender*[Rolle] in Erscheinung. Diese werden in den Kapiteln 5.5 und 5.6 beschrieben.[3]

5.1. Fachexperten

Bei der Entwicklung virtueller Labore bringen die *Fachexperten*[Rolle] ihr Wissen über die zu modellierende Domäne ein. Dieses beinhaltet die theoretischen Grundlagen der Domäne, die umfangreichen Kenntnisse über die Laborausstattung und -benutzung sowie die teilweise sehr komplizierten Experimentabläufe und Versuche, die durchgeführt werden sollen. Ein *Fachexperte*[Rolle] ist demnach also ein Experte der Domäne, die mit dem virtuellen Labor nachgebildet wird.

5.2. Fachdidaktiker

Ein *Fachdidaktiker*[Rolle] ist ein Lehrender an einer Schule oder Universität oder ein Betreuer eines entsprechenden Laborpraktikums. Bei der Entwicklung des virtuellen Labors sorgen die *Fachdidaktiker*[Rolle] für eine didaktische Aufbereitung und Konzeption der fachlichen Inhalte. Dazu ist eine auf die Zielgruppe zugeschnittene Darstellung der durchzuführenden Versuche und der zu erwerbenden Fertigkeiten zu entwickeln. Ein *Fachdidaktiker*[Rolle] ist demnach ein an dem Aufbau und der Umsetzung der Didaktik beteiligter *Projektmitarbeiter*[Rolle].

[1] Auf die Nennung der weiblichen Form der Rollen wird aufgrund der besseren Lesbarkeit verzichtet.

[2] Auf die explizite Einführung einer Rolle des *Didaktikers* für Lehr- und Lernsysteme wird hier verzichtet, weil das *didaktische Konzept*[Artefakt] des virtuellen Labors bereits durch den Workflow des *Tutorkonzeptes*[Workflow] im Wesentlichen vorgegeben wird.

[3] Vergleiche auch die Aufteilung des Produktionsteams für die Entwicklung multimedialer Lernangebote nach (Kerres, 1998, Seite 342) und (Issing, 1997, Seite 207 f.).

5.3. Medienspezialisten

Die *Medienspezialisten*[Rolle] sind für eine ansprechende multimediale Gestaltung des virtuellen Labors verantwortlich und führen sämtliche Aktivitäten der Medienproduktion durch. Allgemein wird daher unter einem *Medienspezialisten*[Rolle] ein an der Medienproduktion beteiligter *Projektmitarbeiter*[Rolle] verstanden. Die *Medienspezialisten*[Rolle] lassen sich noch in weitere Rollen verfeinern. Diese werden in den folgenden Unterkapiteln 5.3.1 bis 5.3.4 beschrieben.

5.3.1. Audio- und Videospezialist

Die *Audio- und Videospezialisten*[Rolle] erstellen und überarbeiten die im virtuellen Labor benötigten Audio- und Videosequenzen. Dabei werden spezielle Werkzeuge zur Verarbeitung der Audio- und Videosequenzen eingesetzt.

5.3.2. Digitalisierer

Die *Digitalisierer*[Rolle] sind für die Erstellung elektronisch zugreifbarer Abbilder von physikalischen Medien mit Hilfe von Scannern zuständig (siehe *Digitalisiere die Medien*[Aktivität] in Kapitel 7.5.5). Physikalische Medien sind zum Beispiel Photos, gedruckte Texte sowie Audio- und Videobänder.

5.3.3. 3D–Modellierungsspezialist

Die *3D–Modellierungsspezialisten*[Rolle] erstellen mit Hilfe spezieller Werkzeuge dreidimensionale Nachbildungen (*3D-Modelle*) realer Laborgeräte, Behälter und Substanzen. Des Weiteren erzeugen die *3D–Modellierungsspezialisten*[Rolle] weitere Medien aus den 3D-Modellen, wie zum Beispiel Graphiken und Animationen.

5.3.4. Graphik-Designer

Die *Graphik-Designer*[Rolle] sind für die Erstellung von Graphiken mit Hilfe entsprechender Werkzeuge für die Bildbearbeitung verantwortlich. Des Weiteren entwerfen die *Graphik-Designer*[Rolle] das Layout der Benutzungsoberfläche und übernehmen das Zusammenfügen der einzelnen Medien (*authoring*).

5.4. Informatiker

Die *Informatiker*[Rolle] übernehmen die Analyse, den technischen Entwurf und die Implementierung des virtuellen Labors. Ein *Informatiker*[Rolle] lässt sich daher als ein

unmittelbar am Software-Entwicklungsprozess beteiligter *Projektmitarbeiter*[Rolle] bezeichnen. Die *Informatiker*[Rolle] lassen sich noch in weitere Rollen unterteilen. Diese werden in den folgenden Unterkapiteln 5.4.1 bis 5.4.8 beschrieben.

5.4.1. Datenbank-Entwickler

Ein *Datenbank-Entwickler*[Rolle] ist für die Planung, den Einsatz und die Anbindung einer Datenbank an das virtuelle Labor verantwortlich. Dazu definiert der *Datenbank-Entwickler*[Rolle] Tabellen, Tabellengrößen, Sichten auf Tabellen, Trigger, gespeicherte Prozeduren (*stored procedures*) und weitere Konstrukte, um Objekte eines virtuellen Labors in einer Datenbank zu speichern, auszulesen und zu löschen (vergleiche Kruchten, 2000, Seite 264).

5.4.2. Designer/Entwickler

Ein *Designer/Entwickler*[Rolle] ist verantwortlich für den Entwurf, die Implementierung und das Testen der Komponenten des virtuellen Labors in Übereinstimmung mit den Richtlinien des Projektes, so dass sich die implementierten Komponenten ohne Probleme in größere Teilsysteme einbetten lassen (vergleiche Kruchten, 2000, Seite 264). Außerdem sind die *Designer/Entwickler*[Rolle] für die Integration der Teilsysteme verantwortlich und schreiben die *technische Dokumentation*[Artefakt] des virtuellen Labors.

5.4.3. Konfigurationsmanager

Der *Konfigurationsmanager*[Rolle] ist verantwortlich für die Bereitstellung der Infrastruktur für das Konfigurationsmanagement und der Arbeitsumgebung für das Entwicklerteam. Zu den Aufgaben des *Konfigurationsmanagers*[Rolle] gehört es also, den *Designern/Entwicklern*[Rolle] eine geeignete Arbeitsumgebung für die Entwicklung und den Test des virtuellen Labors zur Verfügung zu stellen. Außerdem hat der *Konfigurationsmanager*[Rolle] dafür zu sorgen, dass alle Artefakte, die für die Einsatzumgebung benötigt werden, auch verfügbar sind (vergleiche Kruchten, 2000, Seite 264).

5.4.4. Projektleiter

Der *Projektleiter*[Rolle] ist für die Planung, Steuerung und Kontrolle des Projekts verantwortlich (siehe Balzert, 2000b, Seite 60). Zu den Aufgaben des *Projektleiters*[Rolle] gehören die Einteilung der Ressourcen für die Entwicklung des virtuellen Labors, die Festlegung von Prioritäten bezüglich der Aktivitäten und die Überwachung des Projektverlaufs während der Durchführung der Aktivitäten. Des Weiteren koordiniert der *Projektleiter*[Rolle] sämtliche Interaktionen mit dem *Auftraggeber*[Rolle] und sorgt für eine gemeinsame Kommunikationsbasis des Entwicklerteams (vergleiche Kruchten, 2000, Seite 264 f.). Die Rolle des *Projektleiters*[Rolle]

wird während der gesamten Entwicklungsdauer des virtuellen Labors immer durch die gleiche Person besetzt (siehe permanente Rolle in Pasch, 1994, Seite 180).

5.4.5. Simulationsspezialist

Ein *Simulationsspezialist*[Rolle] ist ein *Informatiker*[Rolle], der auf den Entwurf und die Implementierung von Simulationsmodellen in einen konkreten Simulator spezialisiert ist. Der *Simulationsspezialist*[Rolle] muss in der Lage sein, sich sehr gut in die Domäne des virtuellen Labors einzuarbeiten, und arbeitet eng mit den *Fachexperten*[Rolle] zusammen.

5.4.6. System-Administrator

Der *System-Administrator*[Rolle] ist für die Einrichtung und Instandhaltung der Entwicklungsumgebung des virtuellen Labors inklusive der Installation der verwendeten Werkzeuge und Standardkomponenten verantwortlich. Außerdem erstellt der *System-Administrator*[Rolle] in regelmäßigen Abständen Sicherheitskopien von der Entwicklungsumgebung des virtuellen Labors (vergleiche Kruchten, 2000, Seite 265).

5.4.7. System-Analytiker

Der *System-Analytiker*[Rolle] ist für die Erhebung der Anwendungsfälle des virtuellen Labors, also die Erstellung des *Anwendungsfall-Modells*[Artefakt] verantwortlich (vergleiche Kruchten, 2000, Seite 265). Dazu muss der *System-Analytiker*[Rolle] in der Lage sein, sich sehr gut in die Domäne des virtuellen Labors einzuarbeiten. Es ist also ein hoher Kommunikationsbedarf mit den *Fachexperten*[Rolle] und *Fachdidaktikern*[Rolle] vorhanden. Weitere wesentliche Aufgabe der *System-Analytiker*[Rolle] ist die Erstellung und Erweiterung der *Architektur*[Artefakt] des virtuellen Labors (vergleiche die Rolle *architect* in Kruchten, 2000, Seite 263).

5.4.8. Tester

Die *Tester*[Rolle] sind für die Qualitätssicherung des virtuellen Labors verantwortlich. Dazu werden geeignete Testfälle erhoben, zu den Testfällen entsprechende Testprozeduren erstellt und diese von den *Testern*[Rolle] ausgeführt (vergleiche Merx, 1999, Seite 25). Die Testfälle betreffen jedoch nicht die einzelnen Komponenten des virtuellen Labors, die bereits von den *Designern/Entwicklern*[Rolle] getestet werden, sondern beziehen sich ausschließlich auf die einzelnen Teilsysteme und das virtuelle Labor selbst. Außerdem nehmen die *Tester*[Rolle] eine Bewertung der Tests vor (vergleiche Kruchten, 2000, Seite 265).

5.5. Auftraggeber

Der *Auftraggeber*[Rolle] ist der Investor des virtuellen Labors und trägt das finanzielle Risiko der Entwicklung. Außerdem gibt der *Auftraggeber*[Rolle] die Anforderungen an das virtuelle Labor vor. Dies sind zum einen die funktionalen Anforderungen, das heißt eine Beschreibung dessen, was das virtuelle Labor leisten soll, und zum anderen die nicht-funktionalen Anforderungen, wie Vorgaben bezüglich Gestaltung und Layout der Benutzungsoberfläche. Dabei ist die Einhaltung von Vorgaben bezüglich Gestaltung und Layout für den *Auftraggeber*[Rolle] besonders dann von Interesse, falls in dem Unternehmen ein *Corporate Design* definiert ist (siehe Yass, 2000, Seite 88). Des Weiteren erfolgt die Abnahme des virtuellen Labors durch den *Auftraggeber*[Rolle] (siehe Balzert, 2000b, Seite 60). Außerdem kümmert sich der *Auftraggeber*[Rolle] in der Regel auch um den Vertrieb des virtuellen Labors.

5.6. Endanwender

Unter *Endanwender*[Rolle] beziehungsweise Lerner sind die letztendlichen Empfänger des virtuellen Labors zu verstehen (vergleiche Zehnder, 1991, Seite 22). Ein Lerner ist in der Regel eine Person der Zielgruppe, wie zum Beispiel ein Student der entsprechenden Domäne des virtuellen Labors. Es ist aber auch möglich, dass das virtuelle Labor von Lehrenden genutzt wird, die es in der Lehre zur Projektion auf eine Leinwand verwenden und anhand dessen bestimmte Versuchsabläufe erklären.

6. Das Metaobjektmodell für virtuelle Labore im Kontext des VirtLab–Prozess

Das Metaobjektmodell für virtuelle Labore wurde von der Projektgruppe *virtuelle naturwissenschaftlich-technische Labore im Internet* (siehe Projektgruppe Elvis - Internetseiten, 1998) anhand mehrerer real existierender Labore und Praktika erstellt und mittels eines Prototypen getestet. Bei der Erstellung des Metaobjektmodells wurde soweit von den konkreten Ausprägungen existierender Labore und Praktika abstrahiert, dass die wichtigsten Eigenschaften aller Modelle naturwissenschaftlich-technischer Labore in dem Metaobjektmodell festgehalten werden. Damit definiert das Metaobjektmodell eine allgemeine Struktur, die sich durch Labortypunabhängigkeit auszeichnet und durch die Inhalte eines realen Labors oder Praktikums ausgefüllt beziehungsweise erweitert werden kann (siehe Aden u. a., 1999, Seite 35 bis 40). Das Metaobjektmodell ist von zentraler Bedeutung für die Entwicklung virtueller Labore und kann für die Durchführung der verschiedensten Aktivitäten im *VirtLab*–Prozess herangezogen werden. Beispiele hierfür sind die Definition des virtuellen Labors (siehe *Definiere das virtuelle Labor*[Aktivität] in Kapitel 7.2.3), die Analyse der Domäne (siehe *Analysiere die Domäne*[Aktivität] in Kapitel 7.3.2) und die Definition des Frameworks (siehe *Definiere die Architektur*[Aktivität] in Kapitel 7.4.1).

Im Rahmen dieser Arbeit wurde das Metaobjektmodell für virtuelle Labore erneut überprüft, korrigiert und erweitert. So wurden zunächst die Klassennamen korrigiert und der Singular eingeführt. Außerdem wurden die Kardinalitäten der Assoziationen überprüft. Die Aggregationen wurden dahingehend geprüft, ob eine Komposition vorliegt. Zum Schluss wurde das Metaobjektmodell noch um die Abhängigkeiten von Versuch und Teilversuch erweitert. Das so überarbeitete Metaobjektmodell für virtuelle Labore ist als UML-Klassendiagramm in der Abbildung 6.1 dargestellt (vergleiche Aden u. a., 1999, Seite 38).

Zum einen liegt dem Metaobjektmodell die Idee zugrunde, dass im virtuellen Labor vom Lerner interaktive Versuche beziehungsweise Experimente durchgeführt werden können. Zum anderen soll das virtuelle Labor dem Lerner ermöglichen, sich umfassend über die durchzuführenden Versuche und die Laborausstattung zu informieren (siehe Aden u. a., 1999, Seite 37). Diese Denkweise legt nahe, dass Metaobjektmodell zum einen aus der *experimentbezogenen Sicht* (siehe Kapitel 6.1) und zum anderen aus der

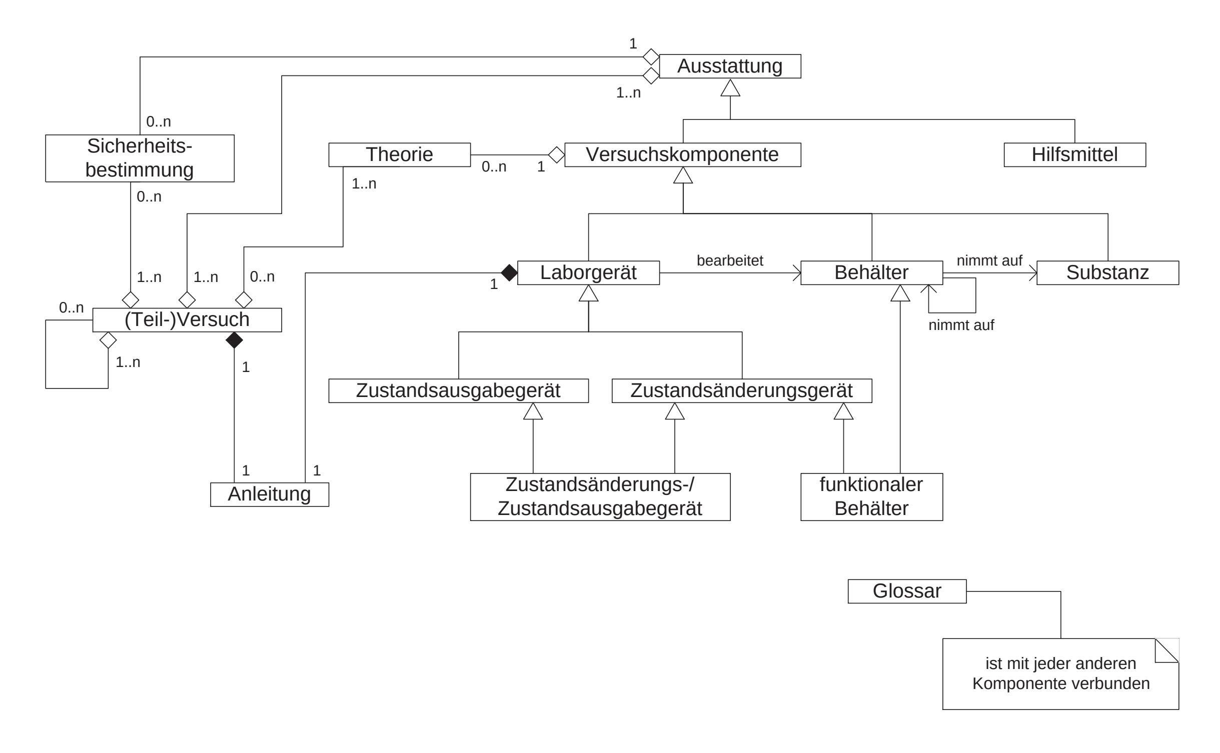


Abbildung 6.1.: Metaobjektmodell für virtuelle Labore

informationsbezogenen Sicht (siehe Kapitel 6.2) zu betrachten. Des Weiteren wird in Kapitel 6.3 auf den Zusammenhang der beiden Sichten eingegangen.

6.1. Experimentbezogene Sicht auf das Metaobjektmodell

Bei der experimentbezogenen Sicht auf das Metaobjektmodell werden alle Klassen und Assoziationen ausgeblendet, die nicht für die Durchführung eines Versuchs benötigt werden, also informierenden Charakter haben. Übrig bleiben alle Klassen und Assoziationen, die für die Modellierung von Versuchen im virtuellen Labor benötigt werden. Die experimentbezogene Sicht auf das Metaobjektmodell ist in der Abbildung 6.2 dargestellt (nach Aden u. a., 1999, Seite 38).

Ein *Versuch* bezeichnet im virtuellen Labor eine bestimmte Handlungsabfolge (Abfolge von bestimmten Arbeitsschritten). Dabei kann ein Versuch aus mehreren *(Teil-)Versuchen* bestehen. Teilversuche[1] können in verschiedenen Versuchen auftauchen oder sind innerhalb eines Versuches mehrfach durchzuführen. In der Regel stellen Teilversuche bestimmte Basistechniken der zu modellierenden Domäne dar, die zur Erreichung wichtiger Zwischenschritte und Teilergebnisse im Versuchsablauf dienen. Dabei ist es möglich, dass die Teilversuche erst noch durch die Angabe konkreter Substanzen instantiiert werden

[1]Entsprechen den in *GenLab* bezeichneten Module.

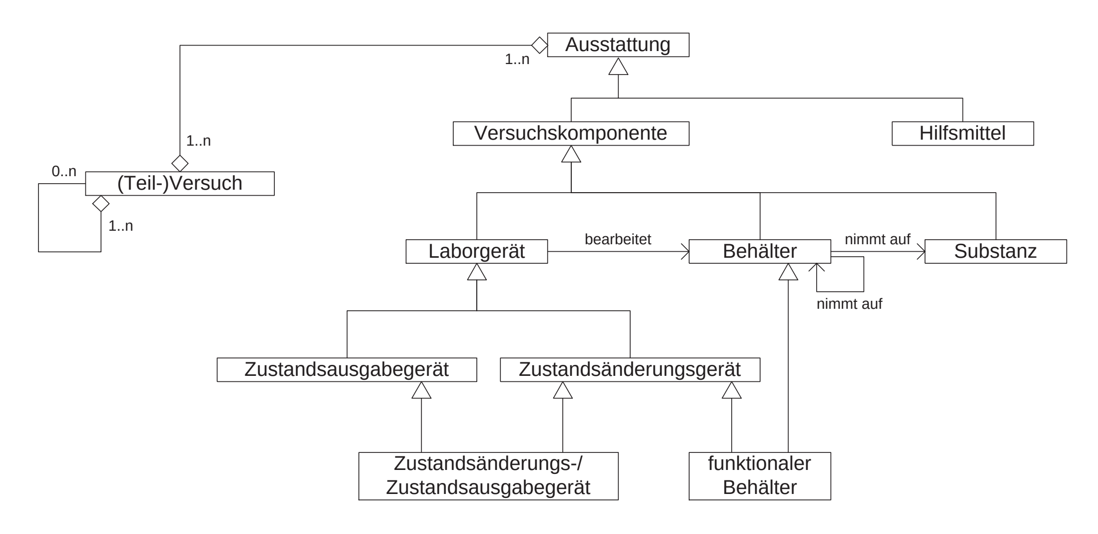


Abbildung 6.2.: Experimentbezogene Sicht auf das Metaobjektmodell

müssen (*generische Teilversuche*). In diesem Fall ist die durch den Teilversuch beschriebene Basistechnik prinzipiell auf verschiedene Substanzen anwendbar. So gibt es im virtuellen Labor *GenLab* zum Beispiel den generischen Teilversuch Restriktion, der in dem Versuch der Restriktionskartierung durch die Angabe eines konkreten Restriktionsenzyms verwendet wird. Des Weiteren können Teilversuche auch als eigenständige Versuche aufgefasst werden und ebenfalls aus weiteren Teilversuchen bestehen. Ein Teilversuch kann aber auch eine geschlossene Einheit bilden, die eine bestimmte *Fertigkeit* im virtuellen Labor darstellt. Besonderes Ziel einer Fertigkeit ist es, dass diese vom Lerner verarbeitet und im Gedächnis gespeichert wird, um sie bei entsprechenden Versuchen wieder abrufen und anwenden zu können (vergleiche Informationsverarbeitung in Strittmatter und Mauel, 1997, Seite 53). Eine Fertigkeit ist zum Beispiel die Bedienung eines einfachen Laborgerätes, wie die Pipette oder Pinzette. Aber auch die Bedienung der Waage oder der korrekte Umgang mit Reagenzgläsern stellt eine Fertigkeit dar. Als allgemeiner Oberbegriff zu Versuch, Teilversuch und Fertigkeit wird auch der Begriff der *Lerneinheit* verwendet (vergleiche IEEE Learning Technology Standards Committee, 2000). Eine Lerneinheit bezeichnet im virtuellen Labor also entweder einen Versuch oder Teilversuch, den es durchzuführen gilt, oder eine Fertigkeit, die zu erwerben ist.

Zur Durchführung von Versuchen im virtuellen Labor wird dem Lerner eine entsprechende Laborausstattung zur Verfügung gestellt. Die Laborausstattung setzt sich aus *Versuchskomponenten* und *Hilfsmitteln* zusammen. Versuchskomponenten sind alle Gegenstände im virtuellen Labor, die für die Durchführung der Versuche notwendig sind. Dagegen müssen Hilfsmittel nicht zwingend vorhanden sein, um einen Versuch durchführen zu können. Beispiele für Hilfsmittel sind Stative, Entsorgungseimer sowie Lagerungs- und Reinigungsgeräte. Aber auch Sicherheitsmittel, wie zum Beispiel ein Feuerlöscher oder ein Notausschalter, sind Hilfsmittel.

Die Versuchskomponenten eines virtuellen Labors lassen sich noch weiter unterteilen in *Laborgeräte*, *Behälter* und *Substanzen*. Das Zusammenspiel der Laborgeräte, Behälter und

Substanzen ist von zentraler Bedeutung für das Verständnis des Simulationsmodells. Daher werden diese drei Klassen im Folgenden detailliert beschrieben (siehe Aden u. a., 1999; Elfreich, 1999):

- Substanzen: Die Substanzen sind die elementaren Bestandteile eines Versuchs. Um das Ziel eines Versuches zu erreichen werden Substanzen bearbeitet und deren Zustände verändert. Die Änderung der Eigenschaften von Substanzen kann durch die Einwirkung anderer Substanzen oder durch Laborgeräte erfolgen. Substanzen können im Allgemeinen nicht direkt bearbeitet werden, sondern nur indirekt über einen Behälter, in oder auf dem sie sich befinden. Beispiele für Substanzen sind die Chemikalien in einem Chemielabor, die elektrischen Bauteile in einem Elektroniklabor oder abstrakter die Daten in einem Software-Labor.

- Behälter: Behälter können Substanzen oder andere Behälter aufnehmen. Des Weiteren umschließen Behälter immer ein gewisses Volumen oder stellen eine Fläche zur Verfügung, auf der Substanzen abgelegt werden können. Beispiele für Behälter die Substanzen aufnehmen können sind Reagenzgläser in einem Chemielabor, Leiterplatten in einem Elektroniklabor und Disketten in einem Software-Labor. Aber auch ein Glasträger zum Mikroskopieren ist ein Behälter. Des Weiteren existieren auch Behälter, die wiederum andere Behälter aufnehmen können. Ein Beispiel ist der Eisbehälter, der Reagenzgläser oder andere kleine Gefäße aufnehmen kann.

- Laborgeräte: Laborgeräte werden dazu eingesetzt, um Zustände von Substanzen zu verändern oder deren momentanen Zustand auszugeben. Prinzipiell lassen sich die Geräte im virtuellen Labor einteilen in *Zustandsänderungsgeräte* und *Zustandsausgabegeräte*. Zustandsänderungsgeräte sind Laborgeräte, die die Zustände von Substanzen verändern und wirken entweder direkt oder indirekt auf die Substanzen ein. Beispiele für Zustandsänderungsgeräte die direkt auf Substanzen einwirken sind die mechanisch-manipulierenden Geräte Schere und Skalpell. Aber auch das thermisch-manipulierende Gerät Mikrowelle wirkt ebenfalls direkt auf die Substanzen und nicht auf die Behälter ein. Beispiel für ein Zustandsänderungsgerät das indirekt auf Substanzen einwirkt, ist der Schüttler zum Schütteln von Substanzen. Zustandsausgabegeräte messen bestimmte Eigenschaften von Substanzen und geben diese aus. Beispiele für Zustandsausgabegeräte sind Thermometer, Waage und Maßband. Aber auch ein Oszilloskop oder ein Strommessgerät stellt ein Zustandsausgabegerät dar.

  Des Weiteren gibt es auch Laborgeräte, die sowohl die Eigenschaften von Zustandsänderungsgeräten als auch von Zustandsausgabegeräten haben und daher als *Zustandsänderungs-/Zustandsausgabegeräte* bezeichnet werden. Beispiele für Zustandsänderungs-/Zustandsausgabegeräte sind der Videorekorder im Multimedia-Labor oder das Diskettenlaufwerk im Software-Labor.

  Laborgeräte haben in der Regel auch Eigenschaften von Behältern, das heißt sie können Substanzen oder Behälter aufnehmen, wie zum Beispiel eine Waage. Ein Zustandsänderungsgerät, das Eigenschaften von Behältern aufweist, wird als *funktionaler Behälter* bezeichnet. Beispiel für einen funktionalen Behälter ist der Kühlschrank.

Dieser kann Behälter mit einem kleineren Volumen und Außenmaß aufnehmen und an den Substanzen, die sich in den Behältern befinden, Zustandsänderungen durch Abkühlung hervorrufen.

Die Assoziationen zwischen den Klassen des Metaobjektmodells geben an, wie diese miteinander in Beziehung treten können. Sie definieren, welche Interaktionen zwischen welchen Komponenten des virtuellen Labors möglich sind, zum Beispiel welche Substanzen auf die Laborgeräte oder auf andere Substanzen in welcher Form reagieren.

6.2. Informationsbezogene Sicht auf das Metaobjektmodell

Analog zu der experimentbezogenen Sicht auf das Metaobjektmodell werden bei der informationsbezogenen Sicht alle Klassen und Assoziationen ausgeblendet, die keinen informierenden Charakter in dem Sinne haben, dass sie dem Lerner dazu dienen, sich ausführlich über die durchzuführenden Versuche und die Laborausstattung zu informieren. Ergebnis der informationsbezogenen Sicht auf das Metaobjektmodell sind die in der Abbildung 6.3 dargestellten Klassen und Assoziationen (nach Aden u. a., 1999, Seite 40).

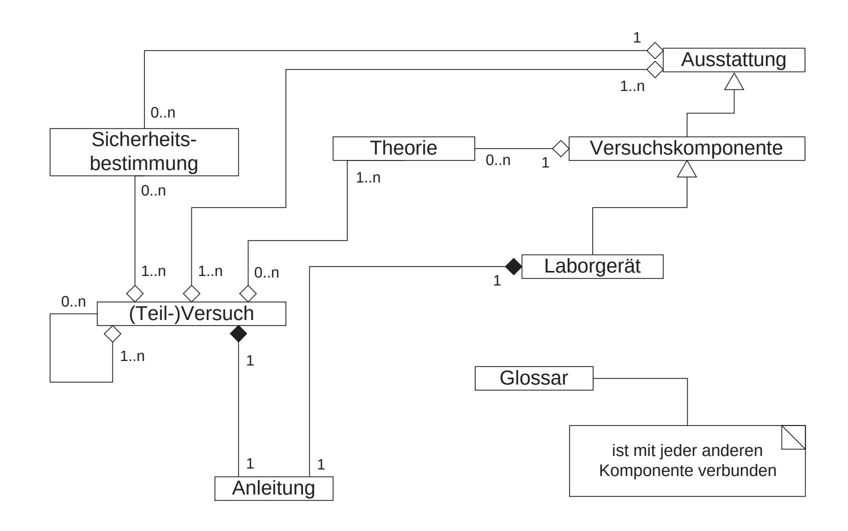


Abbildung 6.3.: Informationsbezogene Sicht auf das Metaobjektmodell

Die Informationsmöglichkeiten, die dem Lerner im virtuellen Labor zur Verfügung stehen, werden durch vier Klassen modelliert. Das sind die *Anleitung*, *Sicherheitsbestimmung*, *Theorie* und das *Glossar*. Anleitungen gibt es sowohl zu jedem Gerät, als auch zu jedem Versuch, der im virtuellen Labor durchgeführt werden kann. Zu jedem Gegenstand, der zur Ausstattung des virtuellen Labors gehört, kann es wichtige Sicherheitsbestimmungen geben. Beispielsweise werden wichtige Hinweise für die richtige Handhabung von Laborgegenständen gegeben, um Unfälle zu vermeiden, oder um die Lebensdauer der

Laborgegenstände zu erhöhen. Sicherheitsbestimmungen sind aber auch bei der Durchführung von Versuchen zu beachten. So ist es beispielsweise denkbar, dass bestimmte Versuche nicht durchgeführt werden dürfen, ohne dass ein Feuerlöscher vorhanden ist. Um ein Experiment durchführen zu können, ist es oft nötig, theoretisches Hintergrundwissen zum Beispiel über Formeln oder Reaktionsgleichungen zu haben. Die Theorie, die hinter Versuchen und deren Ergebnissen steckt, sollte für den Lerner abrufbar sein. Das Glossar ermöglicht schließlich den schnellen und direkten Zugriff auf alle angebotenen Informationen (nach Aden u. a., 1999, Seite 39 f.).

6.3. Zusammenhang der beiden Sichten

Wie stark die experimentbezogene und die informationsbezogene Sicht auf das Metaobjektmodell in der Benutzungsoberfläche des virtuellen Labors miteinander verbunden werden, hängt im Wesentlichen von den Anforderungen des *Auftraggebers*^{Rolle} ab. In *GenLab* zum Beispiel wurden die experimentbezogene und die informationsbezogene Sicht auf das Metaobjektmodell auf der Ebene der Benutzungsoberfläche nahezu vollständig voneinander getrennt. Die experimentbezogene Sicht wurde durch eine *Laborkomponente* realisiert, in der die einzelnen Versuche von *GenLab* durchgeführt werden können (siehe Hasler und Schlattmann, 2001, Seite 7 bis 13). Die informationsbezogene Sicht wurde mit Hilfe einer *Informationskomponente* umgesetzt.[2] Die Informationskomponente in *GenLab* ist ein virtueller Seminarraum mit einer Leinwand, verschiedenen Ordnern und einem Computer mit Zugriff auf das Internet. Die Leinwand dient zum Vorführen von Reaktionsabläufen, die Ordner erlauben dem Lerner einen wahlfreien Zugriff auf das für die Versuche und Versuchskomponenten benötigte Wissen und der Computer steht für eine Internet-Recherche und externe Applikationen bereit (siehe Hasler und Schlattmann, 2001, Seite 13 bis 20). Der Lerner kann über die rechte Maus-Taste beziehungsweise über einen Querverweis (*hyperlink*) von der Laborkomponente zur Informationskomponente wechseln. In der Informationskomponente werden dann die gewünschten Informationen angezeigt (*kontextsensitive Hilfe*). Über eine Schaltfläche in der Informationskomponente gelangt der Lerner wieder zurück in die Laborkomponente.

Neben der konsequenten Aufteilung in Labor- und Informationskomponente wie in *GenLab*, ist es aber auch denkbar, die experimentbezogene Sicht und die informationsbezogene Sicht auf das Metaobjektmodell auch auf der Ebene der Benutzungsoberfläche eng miteinander zu verknüpfen. So können die Informationen zu den Versuchskomponenten oder Reaktionsabläufen des virtuellen Labors, die sonst in der Informationskomponente dargestellt werden, anstelle dessen in einem eigenen Hilfe-Fenster oder *Tooltip* auf der Benutzungsoberfläche der eigentlichen virtuellen Laborumgebung integriert werden.

[2] Achtung: Die in (Hasler und Schlattmann, 2001) verwendeten Begriffe Laborkomponente und Informationskomponente suggerieren, dass es sich hierbei um Komponenten im Sinne der Software-Technik handelt. Die Laborkomponente und die Informationskomponente werden jedoch als Teilsysteme des virtuellen Labors *GenLab* verstanden.

7. Die Workflows, Aktivitäten und Artefakte des VirtLab–Prozess

Für die graphische Darstellung der Workflows wird in dieser Arbeit die Unified Modeling Language (siehe Object Management Group, 1997–2001) verwendet. Diese Entscheidung begründet sich darin, dass der Rational Unified Process als Grundlage für das hier vorgestellte Vorgehensmodell dient und die Unified Modeling Language mit dem Rational Unified Process sehr stark verbunden ist. Die Unified Modeling Language begleitet den objektorientierten Software-Entwicklungsprozess des Rational Unified Process von der Anforderungsbestimmung bis zur Implementierung und wird im Vorgehensmodell selbst zur Darstellung der Workflows verwendet. Zudem ist die Unified Modeling Language momentan der bekannteste Vertreter einer Modellierungssprache für die objektorientierte Software-Entwicklung (Ritter, 2000, Seite 77). Genauer gesagt werden die Workflows mit Hilfe von Aktivitätsdiagrammen der Unified Modeling Language dargestellt (siehe Abbildung 7.1).

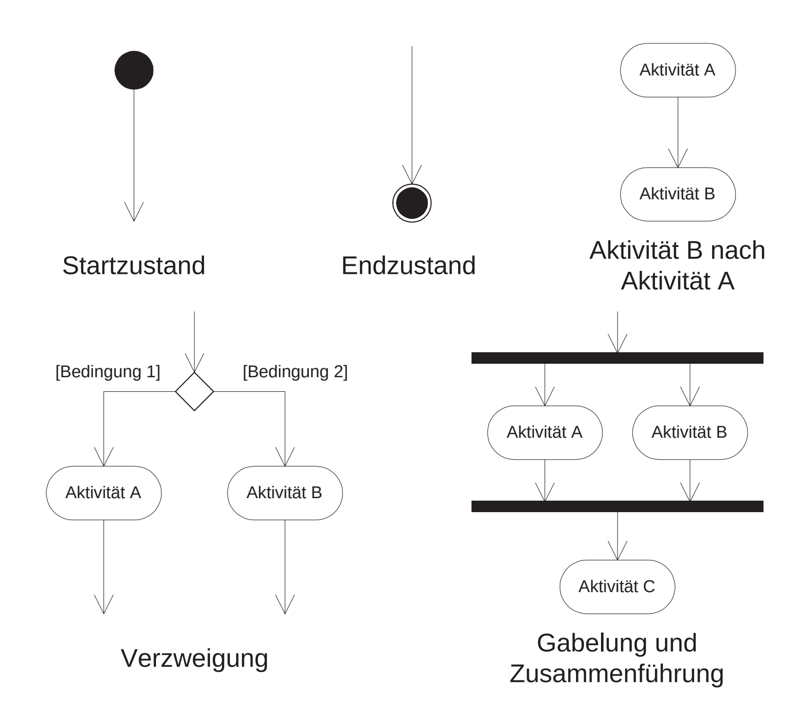


Abbildung 7.1.: Notation zur Darstellung der Workflows

Der Startzustand wird mit einem kleinen schwarzen Kreis und der Endzustand durch das *Bullauge* symbolisiert. Jede einzelne Aktivität wird durch ein Oval dargestellt. Erst wenn eine Aktivität vollständig durchgeführt wurde, erfolgt der Übergang zur nächsten. Die Pfeile beschreiben dabei die Reihenfolge, in der die Aktivitäten ausgeführt werden. Verzweigungen werden mit einer Raute dargestellt und symbolisieren Entscheidungen im Software-Entwicklungsprozess. Dabei wird immer nur genau einer der ausgehenden Pfeile weiter verfolgt. Zur Entscheidungsfindung sind die Ausgangspfeile mit einer Bedingung verknüpft. Gabelungen und Zusammenführungen werden mit einem schwarzen Balken dargestellt. Bei einer Gabelung verzweigt der Kontrollfluss des Software-Entwicklungsprozesses in mehrere parallele Pfade. Eine Gabelung hat immer einen eingehenden und mehrere ausgehende Pfeile. Eine Zusammenführung vereinigt die Kontrollflüsse wieder. Dementsprechend hat die Zusammenführung mehrere eingehende und genau einen ausgehenden Pfeil. Die Aktivität nach einer Zusammenführung, also die am ausgehenden Pfeil, wird erst dann ausgeführt, wenn alle Aktivitäten der eingehenden Pfeile vollständig durchgeführt worden sind (vergleiche Balzert, 2000a, Seite 108 bis 114).

Bei entsprechender Komplexität einer Aktivität wird diese zu einer Detailansicht mit weiteren darin enthaltenen Aktivitäten aufgebrochen. Die graphische Darstellung der Detailansichten folgt der Notation des Rational Unified Process und ist in Abbildung 7.2 dargestellt.

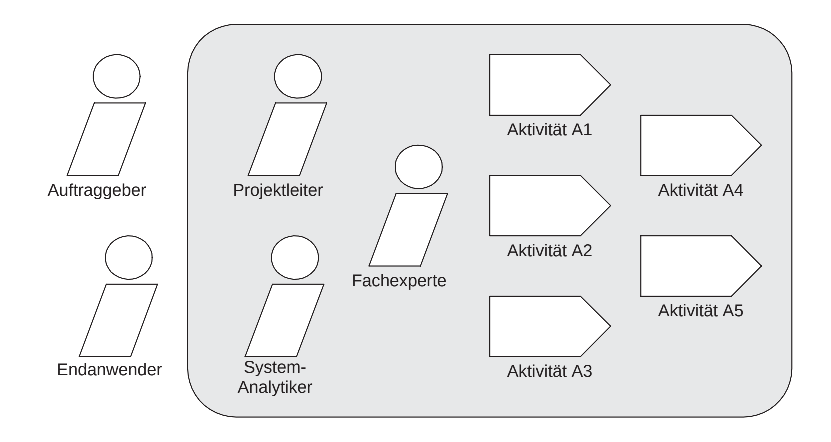


Abbildung 7.2.: Notation zur Darstellung der Detailansichten

Die der aufgebrochenen Aktivität untergeordneten Aktivitäten werden in der Detailansicht jeweils durch einen nach rechts gerichteten Balken dargestellt. Die an der Durchführung der Aktivitäten beteiligten Rollen werden jeweils durch ein nach rechts gezogenes Quadrat mit einem darüber liegendem Kreis dargestellt. Zur Abgrenzung zwischen externen und internen Rollen werden außerdem die Aktivitäten und die internen Rollen durch einen grauen Kasten eingeschlossen. Im Gegensatz zu den Aktivitäten der Workflows stehen bei einer Detailansicht die Aktivitäten in der Regel nicht mehr in einem unmittelbaren logischen oder zeitlichen Zusammenhang. Allerdings sind die Aktivitäten in einer Detailansicht spaltenweise von links nach rechts gewichtet und innerhalb der Spalten von oben nach unten.

Die einzelnen Workflows des *VirtLab*–Prozess werden in den folgenden Kapiteln 7.1 bis 7.10 beschrieben. Die einzelnen Teile der Workflows, das heißt die durchzuführenden Aktivitäten, und die dabei zu erstellenden oder zu verwendenden Artefakte werden in den jeweiligen Unterkapiteln zu den Workflows beschrieben.

7.1. Projektmanagement

Das *Projektmanagement*[Workflow] umfasst die Planung, Organisation, Leitung und Kontrolle zur Entwicklung des virtuellen Labors (siehe Balzert, 2000b; Merx, 1999). Der Workflow des *Projektmanagements*[Workflow] ist identisch mit dem entsprechenden Workflow des Rational Unified Process (siehe Kruchten, 2000, Seite 124) und ist in der Abbildung 7.3 dargestellt.

Während der Definitionsphase wird in der initialen Iteration zunächst die Grundidee für das neue virtuelle Labor erarbeitet. Anschließend wird der Umfang und das Risiko zur Erstellung des virtuellen Labors geschätzt und ein *Pflichtenheft*[Artefakt] inklusive *Gesamtentwicklungsplan*[Artefakt] erstellt. Erscheint die Entwicklung des virtuellen Labors zu teuer oder zu riskant, so wird das Projekt abgebrochen. Wurde der *Gesamtentwicklungsplan*[Artefakt] gebilligt, so beginnt die eigentliche Entwicklung des virtuellen Labors. Nach Erstellung des Entwicklungsplans für die nächste Iteration, wird diese durchgeführt. Gleichzeitig überwacht und kontrolliert der *Projektleiter*[Rolle] den Projektverlauf. Mit Erreichung des Iterationsmeilensteins endet die Iteration. Anschließend wird der restliche Umfang und das verbleibende Risiko bei der weiteren Entwicklung des virtuellen Labors geschätzt. Handelt es sich bei der beendeten Iteration um die letzte Iteration in einer Phase, so wurde ein Phasenende erreicht und die Phase wird als beendet erklärt. Mit der letzten Iteration der Einführungsphase endet schließlich das Projekt.

Die Aktivitäten im *Projektmanagement*[Workflow] werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Ersinne neues virtuelles Labor*[Aktivität] (siehe Kapitel 7.1.1),
- *Schätze Umfang und Risiko des Projekts*[Aktivität] (siehe Kapitel 7.1.2),
- *Erstelle den Gesamtentwicklungsplan*[Aktivität] (siehe Kapitel 7.1.3),
- *Erstelle den Entwicklungsplan für die nächste Iteration*[Aktivität] (siehe Kapitel 7.1.4),
- *Führe die Iteration durch*[Aktivität] (siehe Kapitel 7.1.5),
- *Überwache und kontrolliere den Projektverlauf*[Aktivität] (siehe Kapitel 7.1.6),
- *Beende die Iteration*[Aktivität] (siehe Kapitel 7.1.7),
- *Beende die Phase*[Aktivität] (siehe Kapitel 7.1.8),
- *Beende das Projekt*[Aktivität] (siehe Kapitel 7.1.9) und
- *Schätze restlichen Umfang und Risiko des Projekts*[Aktivität] (siehe Kapitel 7.1.10).

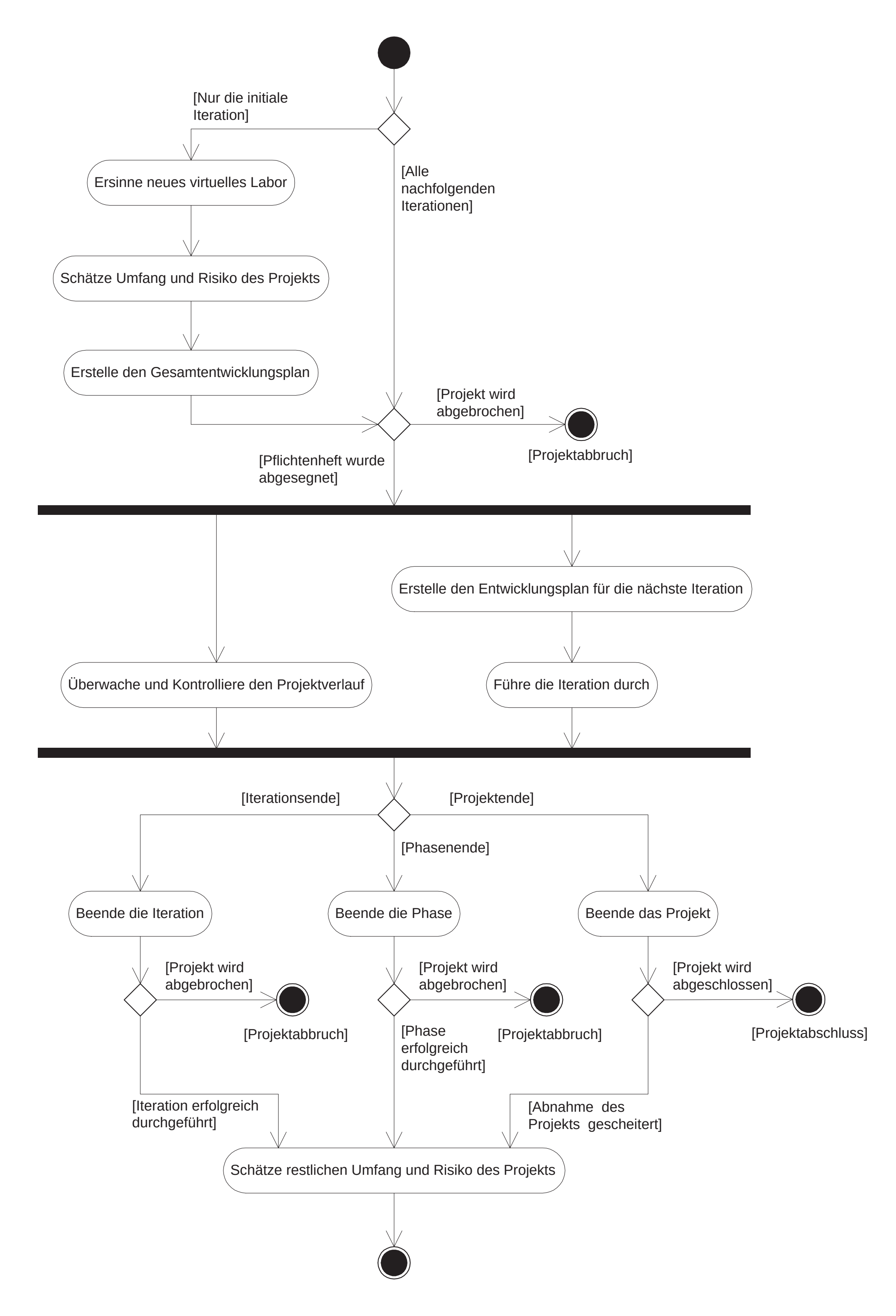


Abbildung 7.3.: Der Workflow für das Projektmanagement

7.1.1. Ersinne neues virtuelles Labor

Soll ein neues virtuelles Labor erstellt werden (siehe Abbildung 7.4), so ist zunächst eine Bedarfsanalyse durchzuführen. Es ist zu prüfen, ob ein virtuelles Labor zu der Domäne überhaupt sinnvoll ist und welche Anforderungen und Wünsche der *Auftraggeber*^Rolle hat. Dazu erarbeiten der *Projektleiter*^Rolle, die *Fachexperten*^Rolle und die *System-Analytiker*^Rolle zusammen mit dem *Auftraggeber*^Rolle das Ziel des virtuellen Labors und erstellen eine initiale Version des *Pflichtenhefts*^Artefakt. Die *Vision* virtueller Labore ist im Allgemeinen, ein bestimmtes Laborpraktikum an einer Universität oder Schule soweit zu ergänzen, dass bei der Durchführung des realen Laborpraktikums nur noch minimal Substanzen und anderes Zubehör verbraucht werden. Um die Vision zu erreichen, kann das Ziel eines virtuellen Labors zum Beispiel sein, dass der Lerner die Handlungsabläufe und Strategien zur Durchführung der Versuche erlernt und die zugehörigen Fertigkeiten erwirbt. Es kann aber auch das Ziel sein, hauptsächlich Strategien zur Durchführung von Versuchen zu vermitteln. Die Fertigkeiten werden dann als bekannt vorausgesetzt und können bei der Durchführung der entsprechenden Versuche übersprungen werden. Neben dem *Pflichtenheft*^Artefakt wird außerdem ein erster *Gesamtentwicklungsplan*^Artefakt erstellt. Dieser enthält lediglich die ungefähre Dauer der einzelnen Phasen und die entsprechenden Phasenmeilensteine und gibt somit einen sehr groben Zeit- und Kostenrahmen für die Entwicklung des virtuellen Labors vor.

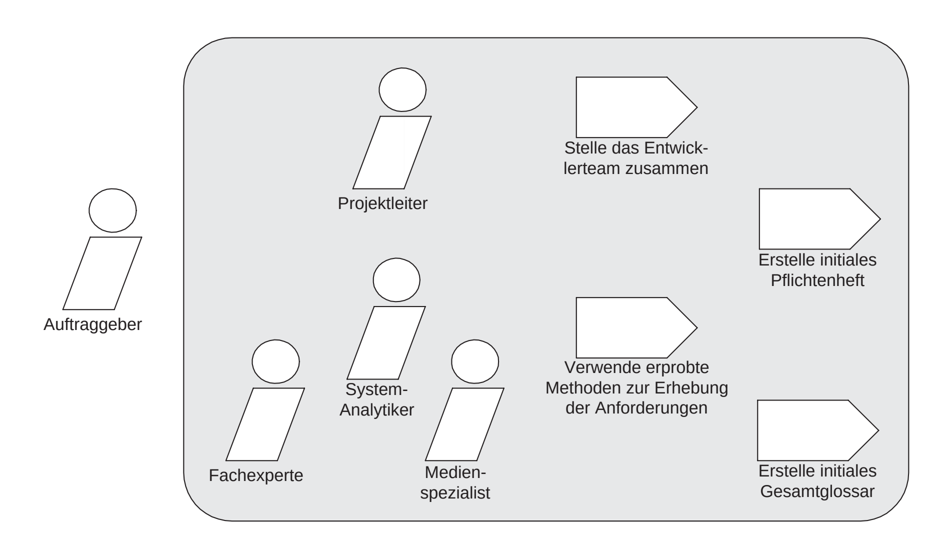


Abbildung 7.4.: Detailansicht der Aktivität Ersinne neues virtuelles Labor

7.1.1.1. Stelle das Entwicklerteam zusammen

In dieser Aktivität stellt der *Projektleiter*^Rolle das Entwicklerteam des virtuellen Labors zusammen. Dieses sollte nicht zu groß sein, um den ohnehin enormen Kommunikationsaufwand wegen der Interdisziplinarität des Entwicklerteams nicht noch weiter zu erhöhen. Ein kleines Entwicklerteam kann untereinander wesentlich effektiver kommunizieren, als ein großes (Sommerville, 1996, Seite 582). Die minimale Größe des Entwicklerteams liegt jedoch bei

drei *Projektmitarbeitern*[Rolle]. Das Entwicklerteam setzt sich dann aus einem *Informatiker*[Rolle], einem *Medienspezialisten*[Rolle] und einem *Fachexperten*[Rolle] zusammen. In diesem Fall ist der *Fachexperte*[Rolle] ein Lehrender (das heißt *Fachdidaktiker*) der Domäne des virtuellen Labors. Die Organisationsform des Entwicklerteams wird in dieser Aktivität noch nicht festgelegt. Diese Überlegungen werden erst in der Aktivität *Erstelle den Gesamtentwicklungsplan*[Aktivität] in Kapitel 7.1.3 durchgeführt.

7.1.1.2. Verwende erprobte Methoden zur Erhebung der Anforderungen

Zur initialen Erhebung der Anforderungen des *Auftraggebers*[Rolle] an das virtuelle Labor sind allgemein bekannte und erprobte Methoden anzuwenden, wie zum Beispiel die Erstellung von *Mind-Maps* (siehe Hertlein, 1997). In Mind-Maps werden die Anforderungen zunächst ohne irgendeinen Kommentar gesammelt und sortiert. Anschließend kann anhand dieser Ideensammlung über die Anforderungen an das virtuelle Labor diskutiert werden. Das ist insbesondere dann hilfreich, wenn der *Auftraggeber*[Rolle] nur eine ungenaue Vorstellung von dem hat, was das virtuelle Labor leisten soll.[1] Eine andere Methode zur Erhebung der Anforderungen ist die in (Merx, 1999, Seite 56 ff.) beschriebene *Qualitätsfunktionen-Darstellung* (QFD, *quality function deployment*). Mit Hilfe der Qualitätsfunktionen-Darstellung können die oft unklaren Anforderungen des *Auftraggebers*[Rolle] in konkrete fachspezifische Informationen umgewandelt werden. Dazu lässt man den *Auftraggeber*[Rolle] zunächst in seiner eigenen Sprache die Anforderungen an das zu entwickelnde Software-System formulieren, bevor man in die Fachsprache wechselt. Zur weiteren Konkretisierung der Anforderungen an das virtuelle Labore ist bei der Qualitätsfunktionen-Darstellung außerdem der Vergleich mit bereits bestehenden Entwicklungen vorgesehen. Dadurch kann das zu entwickelnde virtuelle Labor gegenüber Konkurrenzprodukten abgegrenzt werden (*unique selling proposition*).

7.1.1.3. Erstelle das (initiale) Pflichtenheft

Ein *Pflichtenheft*[Artefakt] beschreibt in knapper Form die wichtigsten Anforderungen, die das zu entwickelnde Software-Produkt aus der Sicht des *Auftraggebers*[Rolle] erfüllen muss (Balzert, 2000a, Seite 229 f.). Zur Erstellung von *Pflichtenheften*[Artefakt] existieren in der Literatur neben allgemeinen Beschreibungen (zum Beispiel Balzert, 2000a,b) auch Vorgehensweisen speziell für multimediale Lehr- und Lernsysteme, wie zum Beispiel (Nagl u. a., 1999, Seite 120 bis 123) und (Schanda, 1995, Seite 177 bis 194). Diese berücksichtigen neben den software-technischen und didaktischen Anforderungen auch die gestalterischen Anforderungen bezüglich der Benutzungsoberfläche virtueller Labore. Daraus wurde ein allgemeines Gliederungsschema des *Pflichtenheftes*[Artefakt] für virtuelle Labore abgeleitet. Dieses ist im Anhang B.1 zu finden. An der Erstellung des *Pflichtenheftes*[Artefakt] sind der *Auftraggeber*[Rolle], der *Fachexperte*[Rolle], die *System-Analytiker*[Rolle], die *Medienspezialisten*[Rolle]

[1] Zur rechnergestützten Erstellung von Mind-Maps kann zum Beispiel der *MindManager* der *Mindjet GmbH* (siehe: http://www.mindjet.de/) verwendet werden.

und der *Projektleiter*[Rolle] beteiligt. Die einzelnen Aktivitäten dieser *Projektmitarbeiter*[Rolle] zur Erstellung des *Pflichtenheftes*[Artefakt] sind in Anhang B.1 als Checkliste zusammengefasst.

7.1.1.4. Erstelle das (initiale) Gesamtglossar

Gleichzeitig zum *Pflichtenheft*[Artefakt] wird ein initiales *Gesamtglossar*[Artefakt] erstellt. In diesem werden alle erklärungsbedürftigen Begriffe und insbesondere die Begriffe der Domäne des virtuellen Labors in alphabetischer Reihenfolge aufgelistet und beschrieben. Das *Gesamtglossar*[Artefakt] fasst dabei alle Einträge aus den Glossaren der einzelnen Dokumente des virtuellen Labors zusammen, mit dem Ziel eine einheitliche Terminologie sicherzustellen (Balzert, 2000b, Seite 69). Die einzelnen Dokumente zum virtuellen Labor sind im Wesentlichen das *Pflichtenheft*[Artefakt], die *Vorstudien*[Artefakt], das *Benutzungshandbuch*[Artefakt] und die *technische Dokumentation*[Artefakt]. Des Weiteren können aber auch die Schablonen und Richtlinien, die während der Entwicklung des virtuellen Labors einzuhalten oder zu berücksichtigen sind, weitere Einträge für das *Gesamtglossar*[Artefakt] enthalten. Das *Gesamtglossar*[Artefakt] wird über den gesamten Software-Entwicklungsprozess des virtuellen Labors weiterentwickelt, das heißt es werden bestehende Einträge korrigiert, gestrichen und neue eingefügt (siehe dazu *Berücksichtige Sprachgebrauch und Denkweisen der Projektmitarbeiter*[Aktivität] in Abschnitt 7.3.2.2).

7.1.2. Schätze Umfang und Risiko des Projekts

Von dem *Projektleiter*[Rolle] und den *System-Analytikern*[Rolle] sind die potentiellen Risiken bei der Entwicklung des virtuellen Labors zu identifizieren und einzuschätzen (siehe Abbildung 7.5). Dazu legen die *System-Analytiker*[Rolle] die Anwendungsfälle des virtuellen Labors fest. Die wichtigsten Anwendungsfälle sind die zur Auswahl und Durchführung der Versuche. Weitere Beispiele für Anwendungsfälle sind das dynamische Laden und Speichern eines Versuchs und der Abruf von Hintergrundinformationen zu einem Versuch oder einer Versuchskomponente (siehe dazu *Erhebe die Anwendungsfälle*[Aktivität] in Abschnitt 7.2.1.1). Anschließend werden die kritischen Anwendungsfälle identifiziert und entsprechende *Vorstudien*[Artefakt] in Auftrag gegeben.

Bekannte und zuverlässige Methode des Risikomanagements ist die Fehlermöglichkeits- und Einflussanalyse (FMEA, *failure mode and effects analysis*). Ziel der Fehlermöglichkeits- und Einflussanalyse ist es, die Fehlerrisiken verschiedener Planungsalternativen zu kalkulieren, um diejenige Vorgehensweise auszuwählen, bei der das Entwicklungsrisiko unter Wahrung der Anforderungen des *Auftraggebers*[Rolle] am geringsten ist. Dazu werden die unterschiedlichen Erfahrungen der *Projektmitarbeiter*[Rolle] bezüglich der kritischen Anwendungsfälle zusammengetragen, um Probleme bei der Entwicklung des virtuellen Labors von vornherein zu vermeiden. Eine wichtige Voraussetzung für den Erfolg der Fehlermöglichkeits- und Einflussanalyse ist daher die Offenheit des Entwicklerteams für gegenseitige Anregungen (siehe Merx, 1999, Seite 66 f.).

Ergebnis dieser Aktivität ist eine vom *Projektleiter*[Rolle] und dem *Auftraggeber*[Rolle] überarbeitete und aktualisierte Fassung des *Pflichtenheftes*[Artefakt] und des *Gesamtentwicklungsplans*[Artefakt]. Dabei fließen sämtliche Überlegungen und Ergebnisse der *Vorstudien*[Artefakt] mit ein.

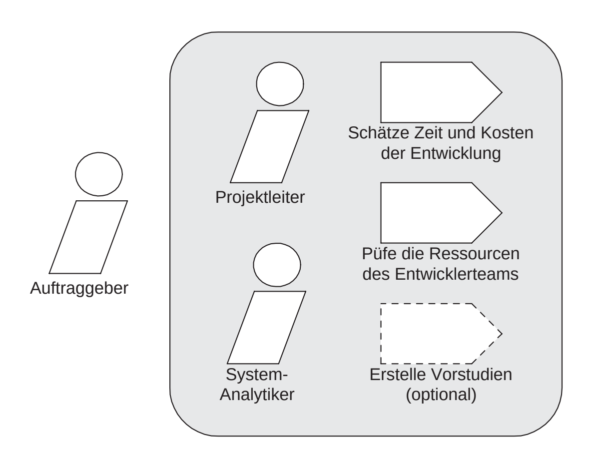


Abbildung 7.5.: Detailansicht der Aktivität Schätze Umfang und Risiko des Projekts

7.1.2.1. Schätze Zeit und Kosten der Entwicklung

Eine zuverlässliche Zeit- und Kostenabschätzung ist für multimediale Lehr- und Lernsysteme und insbesondere für virtuelle Labore nur äußerst schwierig durchzuführen (siehe Nagl u. a., 1999, Seite 182). Vorhandene Aufwandsabschätzungen für klassische Lehr- und Lernsysteme eignen sich hierfür nicht, weil diese nicht präzise genug sind und nur eine äußerst grobe Schätzung der Gesamtkosten angeben, wie zum Beispiel (Heinrich und Schifman, 2000, Seite 114 ff.) und (Nagl u. a., 1999, Seite 189 ff.). Die besonders kostenintensiven Anteile virtueller Labore, das heißt die Animationen, Simulationen und Videosequenzen (siehe Brake, 2000; Issing, 1997) werden nicht berücksichtigt. Generell gilt jedoch, je komplexer die Interaktivität des virtuellen Labors ist, desto teurer wird die Umsetzung mit Hilfe einer höheren Programmiersprache (siehe Sawhney, 1995, Seite 45). Zur Lösung dieser Problematik wird in (Nagl u. a., 1999, Seite 182) vorgeschlagen, etablierte Schätz- und Projektmanagementverfahren, wie zum Beispiel die *Function-Point*-Methode und COCOMO II (siehe Balzert, 2000b, Seite 83 f. und 90 f.), auf die Entwicklung multimedialer Lehr- und Lernsysteme zu übertragen. Problematisch ist dabei jedoch, dass die gestalterischen, didaktischen und inhaltlichen Aspekte bei der Entwicklung virtueller Labore von konventionellen Schätzmethoden nicht berücksichtigt werden. Außerdem müssen die einzelnen Medienanteile wegen der Vielfalt der verwendeten Medien separat geschätzt werden. Dabei kann zum Beispiel folgende Aufteilung vorgenommen werden (vergleiche Nagl u. a., 1999, Seite 187):

- Text nach der Zahl der Worte,
- Graphiken nach Anzahl und Größe, gewichtet nach der Komplexität,

- Animationen nach Laufzeit,
- Audio- und Videosequenzen nach Laufzeit und Herstellungsaufwand und
- die Software-Anteile wie zum Beispiel das Simulationsmodell nach der *Function-Point*-Methode.

Des Weiteren ist die Qualität der Medien zu berücksichtigen. Der Herstellungsaufwand der Medien erhöht sich außerdem je nach Anzahl der Sprachversionen des virtuellen Labors (siehe Nagl u. a., 1999, Seite 122).

7.1.2.2. Prüfe die Ressourcen des Entwicklerteams

Es ist des Weiteren zu prüfen, ob das Entwicklerteam den Anforderungen des virtuellen Labors gerecht wird. So muss ein *Fachexperte*[Rolle] in hinreichendem Maße während der kompletten Entwicklungsdauer des virtuellen Labors zur Verfügung stehen (vergleiche Issing, 1997, Seite 204). Des Weiteren sollte der *Fachdidaktiker*[Rolle] mit der Entwicklung von multimedialen Lehr- und Lernsystemen vertraut sein. Außerdem sollten die *Informatiker*[Rolle] mit den verwendeten Technologien zur Entwicklung des virtuellen Labors hinreichend erfahren sein. Darüber hinaus sollten die *Medienspezialisten*[Rolle] Erfahrung in der Erstellung von realitätsgetreuen Abbildern von Gegenständen der realen Welt haben.

7.1.2.3. Erstelle Vorstudien (optional)

Anhand von *Vorstudien*[Artefakt] wird eine Empfehlung ausgesprochen, ob das geplante Software-Produkt nach Prüfung der fachlichen, personellen und ökonomischen Durchführbarkeit realisiert werden soll (Balzert, 2000b, Seite 69). Eine *Vorstudie*[Artefakt] ist für virtuelle Labore zum Beispiel dann zu erstellen, wenn eine neue Darstellungsart (siehe *Wähle die Darstellungsart*[Aktivität] in Abschnitt 7.2.3.1) oder eine neue Kommunikationstechnologie für die Client/Server-Verteilung (siehe *Verfeinere die Architektur*[Aktivität] in Kapitel 7.4.2) erprobt werden soll. Eine *Vorstudie*[Artefakt] kann entweder einem einzelnen *Projektmitarbeiter*[Rolle] in Auftrag gegeben oder von einer kleinen Gruppe des Entwicklerteams erstellt werden. Bei besonders wichtigen Entscheidungen werden diese gemeinsam von allen *Projektmitarbeitern*[Rolle] des Entwicklerteams diskutiert.

7.1.3. Erstelle den Gesamtentwicklungsplan

Wurde in der Risikoanalyse klar, dass die Entwicklung des virtuellen Labors zu riskant ist, so wird diese Aktivität sehr kurz ausfallen und die Entwicklung abgebrochen. Ansonsten verfeinert der *Projektleiter*[Rolle] den initialen *Gesamtentwicklungsplan*[Artefakt] für das virtuelle Labor (siehe Abbildung 7.6). Dazu werden die Phasen in einzelne Iterationen eingeteilt und entsprechende Iterationsmeilensteine identifiziert (siehe Abschnitt 7.1.3.1). Der *Gesamtentwicklungsplan*[Artefakt] wird des Weiteren während der Entwicklung des virtuellen

Labors nach jeder Iteration aktualisiert. Außerdem wird der *Gesamtentwicklungsplan*[Artefakt] für jede anstehende Iteration um einen detaillierten *Entwicklungsplan der Iteration*[Artefakt] ergänzt.

In dieser Aktivität wird des Weiteren die endgültige Version des *Pflichtenheftes*[Artefakt] erstellt. Dazu muss aus der Aktivität *Analysiere die Domäne*[Aktivität] die *Liste der Lerneinheiten*[Artefakt] (siehe *Analysiere die Domäne*[Aktivität] in Kapitel 7.3.2) vorliegen. Außerdem muss die Definition des virtuellen Labors abgeschlossen sein (siehe *Definiere das virtuelle Labor*[Aktivität] in Kapitel 7.2.3). Zusammen mit dem *Gesamtentwicklungsplan*[Artefakt] stellt das *Pflichtenheft*[Artefakt] den Vertrag mit dem *Auftraggeber*[Rolle] dar. Der Vertrag wird gemeinsam vom *Auftraggeber*[Rolle] und *Projektleiter*[Rolle] unterzeichnet und die Entwicklung des virtuellen Labors kann beginnen (siehe dazu auch Merx, 1999, Seite 109 bis 118).

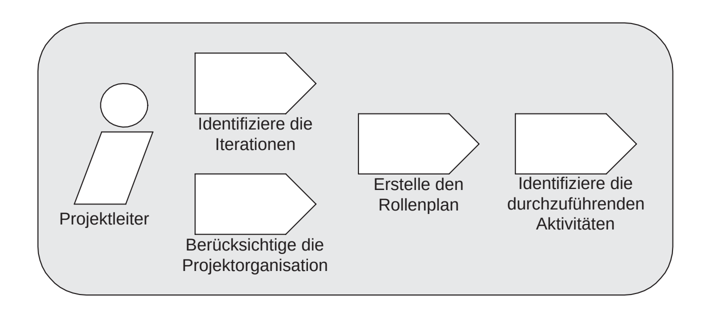


Abbildung 7.6.: Detailansicht der Aktivität Erstelle den Gesamtentwicklungsplan

7.1.3.1. Identifiziere die Iterationen

Für die Entwicklung des virtuellen Labors ist zu klären, wann welche Versuche und Fertigkeiten umgesetzt werden sollen. Des Weiteren ist die Realisierung weiterer Funktionalitäten, wie zum Beispiel das dynamische Laden und Speichern von Versuchen und der Aufruf von Hintergrundinformationen, zu planen. Es bietet sich daher an, den *Gesamtentwicklungsplan*[Artefakt] nach den zu realisierenden Versuchen und Fertigkeiten sowie den weiteren Anwendungsfällen zu strukturieren. Aus den Erfahrungen von *GenLab* heraus, sind für die Umsetzung eines Versuches vier bis sechs Wochen einzuplanen. Dies setzt voraus, dass das Framework zum virtuellen Labor bereits existiert und nur noch ein Teil der Laborgeräte, Behälter und Reaktionsabläufe des Simulators implementiert werden müssen. Für die Umsetzung einer Fertigkeit sind nach den Erfahrungen von *GenLab* ungefähr zwei bis drei Tage einzuplanen, vorausgesetzt die Versuchskomponenten und Reaktionsabläufe sind bereits implementiert. Es bietet sich daher an, die Umsetzung mehrerer Fertigkeiten in einer Iteration zusammenzufassen. Eine typische Iteration in der Entwurfsphase umfasst zum Beispiel die Erstellung eines *annotierten Versuchsprotokolls*[Artefakt] zur detaillierten Spezifikation eines Versuchs oder die Erstellung von *annotierten Versuchsprotokollen*[Artefakt] zur Spezifikation einer bestimmten Anzahl von Fertigkeiten, insofern kein entsprechendes Entwicklungswerkzeug dafür zur Verfügung steht. Das Iterationsziel und damit der Iterationsmeilenstein wird also an der Spezifikation des Versuchs beziehungsweise der Fertigkeiten festgemacht. Eine Iteration kann aber auch aus der

Implementierung, Medienproduktion, Test und Evaluation der Arbeitsflächen, Laborgeräte, Behälter und Substanzen des Versuchs beziehungsweise der Fertigkeiten bestehen. Der Iterationsmeilenstein ist dann die Fertigstellung, das heißt der Abschluss von Tests und Evaluation des Versuchs beziehungsweise der Fertigkeiten. Diese Iteration muss nicht zwangsläufig in der Konstruktionsphase stattfinden, sondern es wird in (Larman u. a., 2001) vielmehr gefordert, dass bereits in der Entwurfsphase insbesondere die *Architektur*[Artefakt] implementiert und getestet wird, da nur so frühzeitig geprüft werden kann, ob auch alle Anforderungen berücksichtigt werden. In der Konstruktionsphase werden dann diejenigen Teile entwickelt, die in der vorherigen Definitionsphase noch nicht berücksichtigt wurden.

7.1.3.2. Berücksichtige die Projektorganisation

Die Organisationsform des Entwicklerteams (*Projektorganisation*) ist für die Planung der Ressourcen von besonderer Bedeutung. Nach (Grupp, 1996, Seite 43) ist für den Aufbau eines interdisziplinären Entwicklerteams die Organisationsform des *Matrix-Projekts* typisch. Hierbei werden die einzelnen *Projektmitarbeiter*[Rolle] entweder als Vollzeit- oder Teilzeit-Mitarbeiter für die Dauer der Entwicklung des virtuellen Labors beschäftigt. Als Vollzeit-Mitarbeiter werden zum Beispiel die *Informatiker*[Rolle] und *Medienspezialisten*[Rolle] eingestellt, während der *Fachexperte*[Rolle] und der *Fachdidaktiker*[Rolle] als Teilzeit-Mitarbeiter an der Entwicklung des virtuellen Labors beteiligt werden. Der *Projektleiter*[Rolle] muss sich dabei ständig um die Verfügbarkeit seiner Teilzeit-Mitarbeiter bemühen, da diese auch an anderen Projekten mitarbeiten. Dabei besteht die Gefahr, dass der *Projektleiter*[Rolle] beginnt Ressourcen zu horten und dadurch anderen Projekten schadet. Außerdem können sich die *Projektmitarbeiter*[Rolle] nicht auf eine Aufgabe konzentrieren (siehe Aquilano u. a., 1998, Seite 54). Die Planung der Ressourcen ist am einfachsten, wenn alle *Projektmitarbeiter*[Rolle], also auch die *Fachexperten*[Rolle] und *Fachdidaktiker*[Rolle], als Vollzeit-Mitarbeiter mit der Entwicklung des virtuellen Labors beschäftigt werden (*pure project*). Der *Projektleiter*[Rolle] hat dann die alleinige Kontrolle über seine *Projektmitarbeiter*[Rolle] und muss diese nicht mit anderen teilen (Aquilano u. a., 1998, Seite 53).

7.1.3.3. Erstelle den Rollenplan

Neben der Projektorganisation ist auch der *Rollenplan* des Entwicklerteams festzulegen. Dazu ist zu entscheiden, welcher *Projektmitarbeiter*[Rolle] während der Entwicklung des virtuellen Labors welche Rollen einnimmt (siehe Pasch, 1994, Seite 180 bis 182). So wird ein *Informatiker*[Rolle] beispielsweise während der Entwurfsphase die Rolle des *System-Analytikers*[Rolle] einnehmen und wechselt in der Konstruktionsphase zum *Designer/Entwickler*[Rolle] und *Tester*[Rolle].

7.1.3.4. Identifiziere die durchzuführenden Aktivitäten

Nach der Bestimmung der Rollen zu den einzelnen *Projektmitarbeitern*[Rolle] über die Zeit, sind die während der Entwicklung des virtuellen Labors durchzuführenden Aktivitäten

zu bestimmen, soweit das hier bereits möglich ist. Für jede Aktivität wird festgelegt, wann diese durchzuführen ist und wie lange die Durchführung dauert. Die Aktivitäten werden des Weiteren bestimmten Rollen, also *Projektmitarbeitern*[Rolle] zugeordnet und im *Gesamtentwicklungsplan*[Artefakt] festgehalten. Die Darstellung der einzelnen Aktivitäten im *Gesamtentwicklungsplan*[Artefakt] und deren Abhängigkeiten voneinander kann zum Beispiel in tabellarischer Form, in Form eines Aktivitätsnetzwerks oder als Balkendiagramm erfolgen (siehe Sommerville, 1996, Seite 54 ff.).

7.1.4. Erstelle den Entwicklungsplan für die nächste Iteration

Der *Projektleiter*[Rolle] bestimmt anhand des *Gesamtentwicklungsplans*[Artefakt] die Inhalte der nächsten Iteration (siehe dazu *Identifiziere die Iterationen*[Aktivität] in Abschnitt 7.1.3.1). Ergebnis dieser Aktivität ist ein detaillierter Entwicklungsplan für die anstehende Iteration (der *Entwicklungsplan der Iteration*[Artefakt]). Dieser beinhaltet eine feingranulare Aufteilung der Aktivitäten, die in der nächsten Iteration durchzuführen sind, und die Zuordnung der Aktivitäten zu den jeweiligen *Projektmitarbeitern*[Rolle]. Ist in der nächsten Iteration eine Medienproduktion vorgesehen, so wird die konkrete Durchführung in der Aktivität *Strukturiere die Medienproduktion*[Aktivität] (siehe Kapitel 7.5.4) geplant. Die Planung der Implementierung wird in der Aktivität *Strukturiere die Implementierung*[Aktivität] in Kapitel 7.6.1 vorgenommen. Anschließend wird die Integration der implementierten Komponenten und der benötigten Medien in der Aktivität *Plane die Integration*[Aktivität] in Kapitel 7.6.2 geplant. Gleichzeitig erfolgt in der Aktivität *Plane die Tests und Evaluation*[Aktivität] (siehe Kapitel 7.7.1) die Planung der Testdurchläufe und der Evaluation des virtuellen Labors. Des Weiteren wird in der Aktivität *Plane den Einsatz*[Aktivität] (siehe Kapitel 7.8.1) der Einsatz des virtuellen Labors vorbereitet. Außerdem wird in der Aktivität *Konzipiere die Entwicklungsumgebung für die Iteration*[Aktivität] (siehe Kapitel 7.10.2) die Entwicklungsumgebung für die anstehende Iteration geplant.

7.1.5. Führe die Iteration durch

Die im *Entwicklungsplan der Iteration*[Artefakt] festgehaltenen Arbeitsanweisungen der aktuellen Iteration werden von den *Projektmitarbeitern*[Rolle] ausgeführt. Am Ende der Iteration wird der Entwicklungsstand des virtuellen Labors vom *Projektleiter*[Rolle] bewertet und im *Ergebnisbericht der Iteration*[Artefakt] festgehalten. Dazu werden die im *Entwicklungsplan der Iteration*[Artefakt] gesteckten Iterationsziele mit den tatsächlich erreichten Zielen verglichen und bewertet. Des Weiteren fließen die Erkenntnisse aus dem *Test- und Evaluationsbericht*[Artefakt] in den *Ergebnisbericht der Iteration*[Artefakt] mit ein.

7.1.6. Überwache und kontrolliere den Projektverlauf

Der *Projektleiter*[Rolle] überwacht und kontrolliert den Projektverlauf während der Durchführung der Iteration anhand des *Entwicklungsplans der Iteration*[Artefakt]. Außerdem äußert der *Projektleiter*[Rolle] eventuelle Änderungswünsche an das virtuelle Labor.

7.1.7. Beende die Iteration

Anhand des *Gesamtentwicklungsplans*[Artefakt] und des *Ergebnisberichts der Iteration*[Artefakt] nimmt der *Projektleiter*[Rolle] eine Bewertung des Projektstatus unter Beachtung der Iterationsmeilensteine vor. Anschließend wird der *Gesamtentwicklungsplan*[Artefakt] aktualisiert und die Iteration als beendet erklärt.

7.1.8. Beende die Phase

Anhand des *Gesamtentwicklungsplans*[Artefakt] und des *Ergebnisberichts der Iteration*[Artefakt] nimmt der *Projektleiter*[Rolle] eine Bewertung des Projektstatus unter Beachtung der Phasenmeilensteine vor. Des Weiteren aktualisiert der *Projektleiter*[Rolle] den *Gesamtentwicklungsplan*[Artefakt]. Anschließend wird die Phase als beendet erklärt.

7.1.9. Beende das Projekt

Die endgültige Version des virtuellen Labors, das heißt das finale *externe Release*[Artefakt] wird an den *Auftraggeber*[Rolle] ausgeliefert. Außerdem wird der Projektstatus von dem *Projektleiter*[Rolle] unter Beachtung der Phasenmeilensteine bewertet und eine letzte Aktualisierung des *Gesamtentwicklungsplans*[Artefakt] vorgenommen. War die Abnahme des virtuellen Labors durch den *Auftraggeber*[Rolle] erfolgreich, so wird das Projekt als beendet erklärt.

7.1.10. Schätze restlichen Umfang und Risiko des Projekts

Diese Aktivität wird jeweils am Ende einer Iteration analog zu der Aktivität *Schätze Umfang und Risiko des Projekts*[Aktivität] (siehe Kapitel 7.1.2) durchgeführt. Die Schätzung erfolgt jedoch in Hinblick auf den restlichen Umfang und Risiko des zu entwickelnden virtuellen Labors. So werden in dieser Aktivität in der Regel keine *Vorstudien*[Artefakt] mehr erstellt. Auch besteht die Aktivität hauptsächlich aus der Aktualisierung und Verfeinerung des *Gesamtentwicklungsplans*[Artefakt].

7.2. Anforderungsbestimmung

Der Workflow der *Anforderungsbestimmung*[Workflow] beinhaltet die Erarbeitung eines gemeinsamen Verständnisses des *Auftraggebers*[Rolle] und des Entwicklerteams über das, was das virtuelle Labor leisten soll. Dazu werden sämtliche Anforderungen des *Auftraggebers*[Rolle] an das virtuelle Labor ermittelt. Außerdem wird die angestrebte Zielgruppe bestimmt. Letztlich dient die *Anforderungsbestimmung*[Workflow] als Grundlage für die Planung der technischen Inhalte und für die Zeit- und Kostenschätzung der einzelnen Iterationen (Kruchten, 2000, Seite 155 f.).

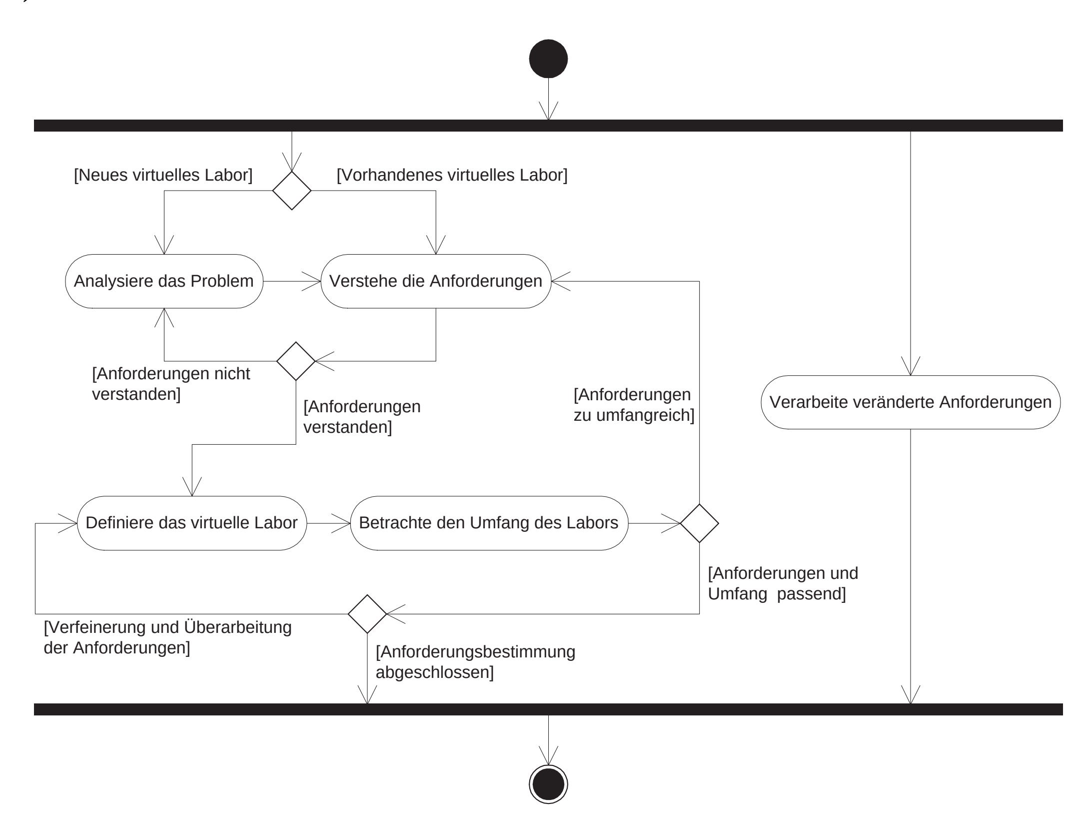


Abbildung 7.7.: Der Workflow für die Anforderungsbestimmung

Die Struktur des in Abbildung 7.7 dargestellten Workflows der *Anforderungsbestimmung*[Workflow] folgt dem Workflow der Anforderungsbestimmung des Rational Unified Process (siehe Kruchten, 2000, Seite 164). Die *Anforderungsbestimmung*[Workflow] teilt sich gleich zu Beginn in zwei parallele Pfade. Im linken Pfad wird der eigentliche Prozess der Anforderungsbestimmung und der Definition des virtuellen Labors durchlaufen. Wird ein neues virtuelles Labor entwickelt, so ist als erstes eine entsprechende Problemanalyse durchzuführen. Wie bei der Weiterentwicklung eines existierenden virtuellen Labors wird anschließend geprüft, ob die Anforderungen des *Auftraggebers*[Rolle] an das virtuelle Labor auch verstanden werden. Unter einer Weiterentwicklung des virtuellen Labors wird dabei auch verstanden, dass das Framework eines bereits existierenden virtuellen Labors wiederverwendet und weiterentwickelt wird. Danach wird das virtuelle Labor definiert,

das heißt es wird festgelegt, was das virtuelle Labor leisten soll. Anschließend wird der Arbeitsumfang für die Entwicklung des virtuellen Labors betrachtet. Werden die Anforderungen als zu umfangreich eingeschätzt, so sind diese zu korrigieren. Der rechte Pfad der *Anforderungsbestimmung*^Workflow^ betrachtet dagegen die sich während der Entwicklung des virtuellen Labors ergebenen Änderungen an den Anforderungen. Diese Änderungen werden verarbeitet und bei der weiteren Entwicklung des virtuellen Labors berücksichtigt.

Die Aktivitäten der *Anforderungsbestimmung*^Workflow^ werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Analysiere das Problem*^Aktivität^ (siehe Kapitel 7.2.1),
- *Verstehe die Anforderungen*^Aktivität^ (siehe Kapitel 7.2.2),
- *Definiere das virtuelle Labor*^Aktivität^ (siehe Kapitel 7.2.3),
- *Betrachte den Umfang des Labors*^Aktivität^ (siehe Kapitel 7.2.4) und
- *Verarbeite veränderte Anforderungen*^Aktivität^ (siehe Kapitel 7.2.5).

7.2.1. Analysiere das Problem

In der Problemanalyse (siehe Abbildung 7.8) ist es besonders wichtig, dass das Entwicklerteam ein gemeinsames Grundverständnis über die Anforderungen und die zu modellierende Domäne des virtuellen Labors findet. Dazu werden die Anforderungen des *Auftraggebers*^Rolle^ an das virtuelle Labor durch den *Projektleiter*^Rolle^ und die *System-Analytiker*^Rolle^ in Form von Anwendungsfällen ermittelt und im *Anwendungsfall-Modell*^Artefakt^ festgehalten. Außerdem wird die Zielgruppe und die Einsatzumgebung bestimmt. Des Weiteren können parallel zu dieser Aktivität die Aktivitäten *Wähle die Wissenserhebungsmethoden*^Aktivität^ und *Analysiere die Domäne*^Aktivität^ des *Tutorkonzeptes*^Workflow^ durchgeführt werden.

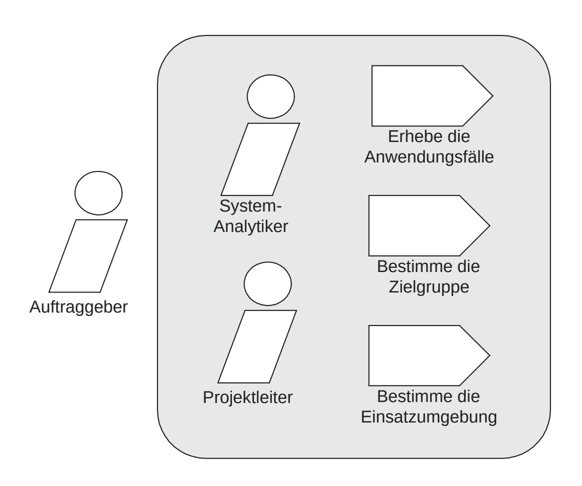


Abbildung 7.8.: Detailansicht der Aktivität Analysiere das Problem

7.2.1.1. Erhebe die Anwendungsfälle

In dieser Aktivität werden von den *System-Analytikern*[Rolle] die Anwendungsfälle des virtuellen Labors erhoben und im *Anwendungsfall-Modell*[Artefakt] festgehalten. Die wichtigsten Anwendungsfälle sind die zur Auswahl und Durchführung eines Versuches (siehe Kapitel 4.1). Es liegt nahe, für jede Lerneinheit des virtuellen Labors, also auch für die (generischen) Teilversuche und Fertigkeiten, einen eigenen Anwendungsfall zu erstellen. Weitere Beispiele für Anwendungsfälle sind der Abruf von Hintergrundinformationen aus einer Informationskomponente und das dynamische Laden und Speichern eines Versuchs.

Für den Anwendungsfall Auswahl und Durchführung eines Versuches kann zur graphischen Darstellung des Versuchsablaufs ein Aktivitätsdiagramm erstellt werden. Dabei beschreibt das Versuchsprotokoll einen konkreten Pfad (Szenario) durch das Aktivitätsdiagramm. Zu einem Versuch gibt es in der Regel mehrere Versuchsabläufe, das heißt es sind alternative Arbeitsabläufe möglich (alternative Szenarien). So können bestimmte Arbeitsschritte im Versuchsprotokoll vertauscht werden, ohne an dem Endergebnis des Versuchs etwas zu ändern. Zur Erstellung eines Aktivitätsdiagramms zu einem Versuch wird das entsprechende Versuchsprotokoll in der Reihenfolge der einzelnen Arbeitsschritte durchgegangen. Arbeitsschritte, die streng nacheinander ausgeführt werden müssen, werden als sequentielle Aktivitäten in das Aktivitätsdiagramm eingetragen. Kann die Reihenfolge von Arbeitsschritten vertauscht werden, so werden diese als parallel ausführbare Aktivitäten in das Aktivitätsdiagramm eingezeichnet. Die Anzahl der möglichen alternativen Versuchsabläufe im Aktivitätsdiagramm sind von der gewünschten Lernform der Lerneinheit abhängig. Bei der freien Experimentiermöglichkeit existieren in der Regel zahlreiche Alternativen. Dagegen beschränkt sich die Anzahl der möglichen Alternativen bei den Fertigkeiten auf einige wenige oder auch nur auf einen einzigen sequenziellen Ablauf (siehe dazu die Aktivität *Bestimme die Navigationsmöglichkeiten in den Lerneinheiten*[Aktivität] in Abschnitt 7.3.3.2). Des Weiteren werden Fehlerfälle, das heißt Aktivitäten die zu einem Fehlerzustand im Simulationsmodell führen, ebenfalls in das Aktivitätsdiagramm eingetragen. Zu jedem Fehlerzustand wird außerdem ein entsprechender Lösungsweg entworfen und in das Aktivitätsdiagramm eingefügt. Die *Fachdidaktiker*[Rolle] überprüfen die Granularität der Arbeitsschritte des Versuchsprotokolls beziehungsweise der Aktivitäten des Aktivitätsdiagramms und verfeinern diese gegebenenfalls. Außerdem werden Vereinfachungen vorgenommen, wo dieses möglich ist. Die *System-Analytiker*[Rolle] vergleichen das Aktivitätsdiagramm mit dem Entwurf und prüfen dabei insbesondere, ob die Funktionalitäten des virtuellen Labors ausreichen oder ob zum Beispiel bestimmte Laborgeräte, Behälter, Substanzen und Reaktionsabläufe im virtuellen Labor noch nicht berücksichtigt werden. Wegen der relativ einfachen Notation der Aktivitätsdiagramme ist es durchaus denkbar, dass diese Darstellungsform der Anwendungsfälle nicht nur von den *Informatikern*[Rolle], sondern auch von den *Fachexperten*[Rolle] und *Fachdidaktikern*[Rolle] verstanden und akzeptiert wird. So können bei entsprechender Unterstützung durch ein Entwicklungswerkzeug auch *Fachexperten*[Rolle] oder *Fachdidaktiker*[Rolle] die Aktivitätsdiagramme zu den Versuchen erstellen. Ein Beispiel eines Anwendungsfalls zur Auswahl und Durchführung eines Versuchs dargestellt als Aktivitätsdiagramm ist im Anhang A zu finden.

Neben Aktivitätsdiagrammen werden zur Beschreibung des Anwendungsfalls zur Auswahl und Durchführung eines Versuchs auch Zustandsdiagramme erstellt. Diese beschreiben welche Zustände die verwendeten Laborgeräte und Behälter haben können und legen damit fest, wie sich die Laborgeräte und Behälter in die Aktivitäten des Aktivitätsdiagramms integrieren. Dabei wird mit dem Zustandsdiagramm keine technische Modellierung der Laborgeräte und Behälter vorgenommen, sondern es werden nur soweit diskrete Zustände identifiziert und modelliert, wie sie zur Durchführung des Versuchs notwendig sind. Des Weiteren können die Zustandsdiagramme zu den Laborgeräten und Behältern für andere Aktivitätsdiagramme wiederverwendet werden. Beispiele für Zustandsdiagramme befinden sich im Anhang A.

Des Weiteren kann der Aufbau eines Versuchs mit Hilfe eines Klassendiagramms und dabei insbesondere mit den Aggregationen zwischen den Arbeitsflächen und den Versuchskomponenten beschrieben werden. Zur detaillierteren Darstellung der dynamischen Abläufe im virtuellen Labor können außerdem noch Sequenzdiagramme zu den Anwendungsfällen erstellt werden (Balzert, 2000a, Seite 132 ff.).

7.2.1.2. Bestimme die Zielgruppe

In der Problemanalyse wird des Weiteren die Zielgruppe des zu entwickelnden virtuellen Labors bestimmt. Allgemein kommen für den Anwendungsbereich virtuelle Labore sämtliche Personengruppen und Altersklassen ab dem ersten Schuljahr als Zielgruppe in Frage. In (Kerres, 1998, Seite 144) wird vorgeschlagen, die Zielgruppe anhand der soziodemographischen Daten zu charakterisieren, wie:

- die Größe der Zielgruppe, das heißt die maximale und minimale Anzahl potentieller Benutzer sowie die erwartete Anzahl von Benutzern,
- die geographische Verteilung bezüglich Regionen und Nationen,
- das Alter im Durchschnitt und als Spanne,
- die Verteilung des Geschlechts,
- der höchste schulische Abschluss,
- die Benutzergruppe (betriebliche Anwender oder Heim-Anwender) sowie
- die Kaufbereitschaft der Benutzergruppe.

Die soziodemographischen Daten werden hauptsächlich für die Planung der Distribution des virtuellen Labors erhoben. Für die Erstellung des *didaktischen Konzeptes*[Artefakt] sind sie dagegen nur von geringer Bedeutung.

7.2.1.3. Bestimme die Einsatzumgebung

Neben den soziodemographischen Daten ist bei der Bestimmung der Zielgruppe auch die Einsatzumgebung von wichtiger Bedeutung, das heißt die verfügbaren und voraussetzbaren Ressourcen an die Computer-Systeme der *Endanwender*[Rolle], auf denen das virtuelle Labor später ablaufen soll. Für die Bestimmung der Einsatzumgebung sind folgende vier Kriterien zu betrachten:

- *Software*: Es ist die Zielplattform, das heißt das Betriebssystem auf der das virtuelle Labor später ablaufen soll, zu bestimmen. Auch ist zu ermitteln, ob und welche Gerätetreiber auf der Zielplattform vorausgesetzt werden müssen, zum Beispiel um im virtuellen Labor Videosequenzen abspielen oder 3D-Modelle darstellen zu können.

- *Hardware*: Bezüglich der Hardware ist zu bestimmen, welche Anforderungen an die Computer-Systeme der *Endanwender*[Rolle] gestellt werden können beziehungsweise dürfen. Des Weiteren ist eine Minimalanforderung zu ermitteln, das heißt es ist festzulegen, welche Anforderungen die Hardware erfüllen muss, damit das virtuelle Labor überhaupt genutzt werden kann.

- *Orgware*: Es ist zu prüfen, ob bei den *Endanwendern*[Rolle] ein Zugang zum Internet vorausgesetzt werden kann oder nicht. Ist dies der Fall, so ist die minimale, maximale und durchschnittliche Geschwindigkeit der Internetanbindung zu bestimmen.

- *Teachware*: Es ist festzulegen, ob dem virtuellen Labor entweder ein umfangreiches gedrucktes *Benutzungshandbuch*[Artefakt] (Begleitmaterial) beigelegt werden soll, oder ob ein elektronisches *Online-Handbuch* zu erstellen ist (siehe dazu *Erstelle das Benutzungshandbuch*[Aktivität] in Abschnitt 7.8.2.1).

Neben der Festlegung der Zielgruppe ist die Bestimmung der Einsatzumgebung auch für den späteren Vertrieb des virtuellen Labors von wichtiger Bedeutung (siehe dazu *Vertreibe das virtuelle Labor*[Aktivität] in Kapitel 7.8.6). Typische Zielgruppen für virtuelle Labore sind zum Beispiel Schüler jeder Alters- und Klassenstufe, Studenten der naturwissenschaftlich-technischen Fächer sowie Auszubildende eines handwerklich-orientierten Berufs.

7.2.2. Verstehe die Anforderungen

Es wird vom *Projektleiter*[Rolle] geprüft, ob alle Anforderungen des *Auftraggebers*[Rolle] an das virtuelle Labor auch von den *System-Analytikern*[Rolle], *Fachexperten*[Rolle], *Fachdidaktikern*[Rolle] und *Medienspezialisten*[Rolle] korrekt verstanden werden. Des Weiteren prüft der *Projektleiter*[Rolle], ob die Bedürfnisse aller *Projektmitarbeiter*[Rolle] erfüllt werden, das heißt ob alle mit den Anforderungen und dem geplanten Vorgehen zur Realisierung des virtuellen Labors einverstanden sind.

7.2.3. Definiere das virtuelle Labor

Die Definition des virtuellen Labors (siehe Abbildung 7.9) wird von den *System-Analytikern*Rolle durchgeführt und umfasst sämtliche Entscheidungen, die die Modellierung des virtuellen Labors betreffen. Sie ist eine der wesentlichen Aktivitäten bei der Erstellung des virtuellen Labors, da sich die hier getroffenen Entscheidungen sowohl auf das *didaktische Konzept*Artefakt (siehe *Erstelle das didaktische Konzept*Aktivität in Kapitel 7.3.3), die *Architektur*Artefakt (siehe *Definiere die Architektur*Aktivität in Kapitel 7.4.1) als auch auf das Layout und die Gestaltung der Benutzungsoberfläche (siehe *Erstelle das Gesamtdesign*Aktivität in Kapitel 7.5.1) des virtuellen Labors auswirken.

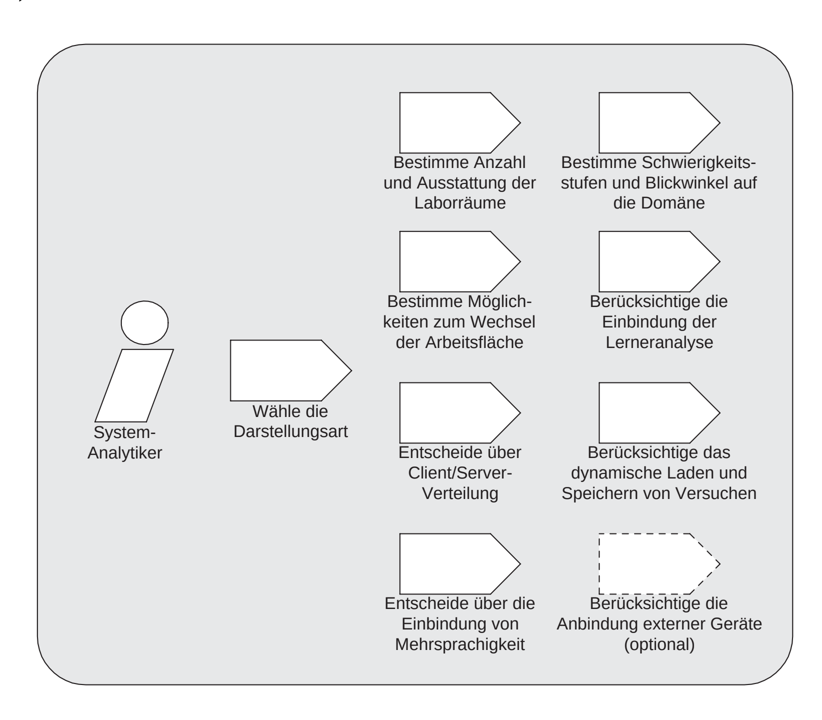


Abbildung 7.9.: Detailansicht der Aktivität Definiere das virtuelle Labor

7.2.3.1. Wähle die Darstellungsart

Als erste Entscheidung bei der Definition des virtuellen Labors wird festgelegt, welche *Darstellungsart* für die Benutzungsoberfläche verwendet werden soll. Prinzipiell wird hier zwischen den folgenden beiden Darstellungsarten unterschieden:

- *2D-Darstellung*: In der 2D-Darstellung wird das virtuelle Labor mit Hilfe von zweidimensionalen Graphiken auf dem Bildschirm dargestellt. Die Graphiken können zum Beispiel mit einem entsprechenden Werkzeug erstellt oder zuvor aus einem 3D-Modell generiert worden sein. Um einen gewissen Tiefeneindruck bei dem Lerner zu erzeugen, kann die 2D-Darstellung um eine *Z-Sortierung* (siehe *Konzipiere Mechanismen der explorativen Lehr- und Lernumgebung*Aktivität in Abschnitt 7.4.1.4)

ergänzt werden (vergleiche Hasler und Schlattmann, 2001, Seite 10). Die 2D-Darstellung ist derzeit Stand der Entwicklung virtueller Labore und wird beispielsweise in *GenLab* (siehe GenLab - Internetseiten, 2001) und der *Virtual Physiology Reihe* (siehe Virtual Physiology - Internetseiten, 2001) eingesetzt.

- *3D-Darstellung*: Bei der 3D-Darstellung werden hauptsächlich 3D-Modelle für die Darstellung des virtuellen Labors verwendet. Die 3D-Modelle werden in Echtzeit berechnet und auf dem Bildschirm dargestellt. Bei der 3D-Darstellung werden daher im Gegensatz zu der 2D-Darstellung erheblich höhere Anforderungen an die Ressourcen des verwendeten Computer-Systems gestellt. Dafür ist mit der 3D-Darstellung eine höhere Flexibilität in der Darstellung der Laborgegenstände möglich. Ein prototypischer Lösungsansatz für die 3D-Darstellung wurde zum Beispiel in (Heuten, 2001) für das virtuelle Labor *GenLab* realisiert.

Für beide Darstellungsarten gilt, dass deren Entwicklung recht aufwendig ist. Welche Darstellungsart gewählt wird, hängt daher im Wesentlichen von den Anforderungen des *Auftraggebers*^Rolle und den voraussetzbaren Rechnerressourcen bei den *Endanwendern*^Rolle ab. Weiteres Problem bei der 3D-Darstellung ist die Navigation im dreidimensionalen Raum mit Hilfe der Computer-Maus, die eigentlich nur für eine zweidimensionale Navigation konzipiert ist. Die Labornavigation kann dann zum Beispiel durch die Tastatur ergänzt und vervollständigt werden (siehe Heuten, 2001).

7.2.3.2. Bestimme Anzahl und Ausstattung der Laborräume

Wichtig für die Bestimmung der *Anzahl und Ausstattung der Laborräume* ist die Entscheidung, ob ein reales Labor auf dem Computer-System abgebildet werden soll, oder ob eine fiktive virtuelle Laborumgebung zu erstellen ist. Soll ein reales Labor *eins-zu-eins* auf einem Computer-System nachgebildet werden, wie zum Beispiel bestimmte Laborräume der Carl von Ossietzky Universität Oldenburg, dann kommt es besonders auf eine realitätsgetreue Nachbildung der realen Laborräume und Arbeitsflächen an. Die Anzahl der Laborräume und Arbeitsflächen im virtuellen Labor entspricht im Idealfall also der Anzahl im realen Labor (siehe dazu unten). Des Weiteren müssen sich sämtliche Laborräume, Arbeitsflächen und die darauf befindlichen Versuchskomponenten im virtuellen Labor an derselben Stelle befinden, wie im realen Labor auch. Außerdem sollten die Laborräume, Arbeitsflächen und Versuchskomponenten falls möglich photorealistisch dargestellt werden. Dadurch entsteht ein hoher Wiedererkennungseffekt für den Lerner. Dieser findet sich nach der Vorbereitung im virtuellen Labor sofort in der realen Laborumgebung zurecht (siehe dazu auch *Erstelle das Gesamtdesign*^Aktivität in Kapitel 7.5.1).

Soll dagegen ein fiktives virtuelles Labor bezüglich einer bestimmten Domäne erstellt werden, ohne dass die Räume und Arbeitsflächen sich an ein konkretes reales Labor orientieren, wie zum Beispiel das virtuelle Genlabor *GenLab*, so müssen zunächst mehrere reale Labore der Domäne analysiert werden. Anschließend werden die Gemeinsamkeiten und Unterschiede der realen Labore identifiziert, das heißt es werden alle denkbaren

beziehungsweise möglichen Laborräume und Arbeitsflächen aufgelistet und davon diejenigen bestimmt, welche für die Durchführung der Versuche im virtuellen Labor benötigt werden. Für die Erstellung der fiktiven Laborumgebung ist es also wichtig, dass eine sinnvolle Schnittmenge aus den bestehenden Laboren der Domäne auf dem Computer-System abgebildet wird. Es kommt dann nicht darauf an, dass sich der Lerner nach der Vorbereitung im virtuellen Labor anschließend sofort in einem realen Labor zurecht findet, sondern dass er die im virtuellen Labor erlernten Konzepte und Handlungsabläufe versteht und auf ein beliebiges reales Labor übertragen kann. Im Fall des fiktiven virtuellen Labors sind die Arbeitsflächen und Laborräume so im virtuellen Labor zu verteilen, dass bei der Durchführung der Versuche minimale Wegstrecken zurückgelegt werden.

Unabhängig davon, ob ein reales Labor oder ein fiktives Labor abgebildet werden soll, besteht ein virtuelles Labor immer mindestens aus einem Laborraum, der wiederum mindestens eine Arbeitsfläche enthält. Des Weiteren kann ein virtuelles Labor prinzipiell eine beliebige Anzahl an Laborräumen und Arbeitsflächen enthalten. Besteht das virtuelle Labor aus mehr als einer Arbeitsfläche, so muss entschieden werden, ob eine Transportleiste[2] benötigt wird, um Laborgegenstände zwischen den einzelnen Arbeitsflächen transportieren zu können. Wie viele Arbeitsflächen letztendlich benötigt werden, ist abhängig von der Domäne des virtuellen Labors und der Komplexität sowie dem Platzbedarf der Versuche. Die dabei auftretenden Probleme werden in der Aktivität *Spezifiziere Aufbau und Ablauf der Lerneinheit*^Aktivität (siehe Kapitel 7.3.4) beschrieben.

7.2.3.3. Bestimme Möglichkeiten zum Wechsel der Arbeitsfläche

Besteht das virtuelle Labor aus mehr als einer Arbeitsfläche, so muss die Möglichkeit zum *Wechsel der Arbeitsfläche* bestehen. Dabei sind folgende Möglichkeiten des Arbeitsflächenwechsels beziehungsweise der Navigation im virtuellen Labor denkbar:

- *Menüsteuerung*: Der Wechsel zwischen den einzelnen Arbeitsflächen des virtuellen Labors erfolgt über ein integriertes Auswahlmenü. Mit Hilfe der Menüsteuerung kann relativ einfach zwischen den Arbeitsflächen im virtuellen Labor gewechselt werden. Allerdings ist die vermittelte Realitätsnähe bei alleiniger Verwendung der Menüsteuerung sehr gering.

- *Grundrissansicht*: In der Grundrissansicht wird das virtuelle Labor so dargestellt, dass die einzelnen Arbeitsflächen identifiziert werden können. Der Wechsel zwischen den Arbeitsflächen erfolgt durch Auswahl mit der Computer-Maus (siehe Hasler und Schlattmann, 2001, Seite 9).

- *Flickenteppich*: Die einzelnen Arbeitsflächen des virtuellen Labors werden so aneinander gereiht, dass der Eindruck einer großen Arbeitsfläche (ein sogenannter *Flickenteppich*) entsteht. Im virtuellen Labor wird dann immer nur ein Ausschnitt der großen Arbeitsfläche dargestellt. Der Lerner kann zwischen den einzelnen

[2] Siehe Glossar beziehungsweise Kapitel 2.8

Arbeitsflächen entweder durch einen Rollbalken (*scrollbar*) wechseln, oder die Computer-Maus direkt zur nächsten Arbeitsfläche ziehen. Dabei ändert sich der dargestellte Ausschnitt der großen Arbeitsfläche entsprechend der Bewegung des Rollbalkens beziehungsweise der Computer-Maus.

- *3D-Raummetapher*: Die 3D-Raummetapher (siehe Kerres, 1998, Seite 255) imitiert eine echte 3D-Umgebung und besteht aus einer Menge von *Einzelansichten* des virtuellen Labors. Eine Einzelansicht ist eine vorgefertigte Graphik, die einen bestimmten Ausschnitt des virtuellen Labors darstellt. Zwischen den Einzelansichten ist ein Beziehungsnetzwerk definiert. Dieses legt fest, wann und wie zwischen welchen Einzelansichten des virtuellen Labors gewechselt werden kann (vergleiche die Umsetzung der Labornavigation in Hasler und Schlattmann, 2001, Seite 9 f.). Bei einer genügend hohen Anzahl von zuvor generierten Einzelbildern wird beim Lerner der Eindruck einer kontinuierlichen Bewegung durch ein dreidimensionales virtuelles Labor erzeugt (vergleiche Kerres, 1998, Seite 255).

- *Virtuelle Realität* (VR, *virtual reality*): Die Navigation im virtuellen Labor erfolgt bei der virtuellen Realität (siehe Alsdorf und Bannwart, 1997, Seite 438 f.) in einer echten 3D-Umgebung, das heißt auf der Basis eines formalisierten 3D-Modells (vergleiche Heuten, 2001, Seite 15 f.). Bei der virtuellen Realität werden die Einzelbilder des virtuellen Labors inklusive aller Arbeitsflächen und Versuchskomponenten in Echtzeit generiert (vergleiche Kerres, 1998, Seite 255). Die virtuelle Realität stellt also eine konsequente Weiterführung der zuvor genannten 3D-Raummetapher dar.

Die Konzepte zum Wechsel der Arbeitsflächen schließen sich nicht gegenseitig aus, sondern ergänzen sich unter Umständen sogar. So ist in *GenLab* eine Kombination aus Menüsteuerung und 3D-Raummetapher zu finden. Um Laborgeräte und Substanzen zwischen den einzelnen Arbeitsflächen im virtuellen Labor transportieren zu können, ist zum Beispiel die Integration einer Transportleiste in die Arbeitsflächenverwaltung denkbar, wie zum Beispiel in *GenLab* realisiert (siehe Hasler und Schlattmann, 2001, Seite 13).

7.2.3.4. Entscheide über Client/Server-Verteilung

Soll das virtuelle Labor als *Einzelplatzanwendung* konzipiert werden, so ist keine Client/Server-Verteilung vorzunehmen. Bei einer Internetanwendung ist dagegen eine geeignete Client/Server-Verteilung festzulegen. Dabei kommen allerdings nur *client*seitige Techniken in Frage, bei der sich die eigentliche Darstellung des virtuellen Labors und die komplette Interaktion mit dem Lerner auf dem Client befindet (siehe Balzert, 2000a, Seite 177). Nur so kann die geforderte hohe Interaktivität virtueller Labore erreicht werden. Handelt es sich bei dem virtuellen Labor um eine Internetanwendung, so arbeitet jeder Client beziehungsweise Lerner mit einer eigenen Instanz des virtuellen Labors, auf die die anderen Lerner nicht zugreifen können. Auch haben die Lerner in der Regel keine Kenntnis über die Existenz weiterer Instanzen, das heißt die Lerner wissen nicht, ob und wie viele andere Lerner gerade mit dem virtuellen Labor arbeiten.

Es ist aber ebenso denkbar, dass im sogenannten *Mehrbenutzerbetrieb* mehrere Lerner gleichzeitig auf eine gemeinsame Instanz des virtuellen Labors arbeiten. In diesem Zusammenhang ist auch die Unterstützung weiterer Arbeitsformen beziehungsweise Lernformen zu betrachten. Denkbar ist zum Beispiel kooperatives Lernen, bei dem mehrere Lerner gemeinsam ein Problem im virtuellen Labor lösen (siehe dazu Hesse u. a., 1997). Dies kann zum Beispiel in Zweiergruppen erfolgen. Aber auch kompetitives Lernen, also das Lernen im gegenseitigen Wettbewerb, ist denkbar.

Des Weiteren können auch *verteilte virtuelle Labore* realisiert werden. Dabei greift ein Client auf mehrere Server zu. So kann ein Versuchsablauf von einem ersten Server geladen werden, während für die Berechnung von komplexen Reaktionsabläufen auf einen weiteren Server zurückgegriffen wird.

7.2.3.5. Entscheide über die Einbindung von Mehrsprachigkeit

Es ist zu entscheiden, ob das virtuelle Labor *Mehrsprachigkeit* unterstützen soll. Mehrsprachigkeit bedeutet allgemein, dass dem Lerner mehrere Sprachen für das virtuelle Labor zur Auswahl stehen. Bei der Darstellung des virtuellen Labors sind also die Besonderheiten der jeweiligen Sprache zu berücksichtigen (siehe auch *Ermittle benötigte Medien*^Aktivität in Kapitel 7.5.2).

Soll Mehrsprachigkeit nicht gleich zu Beginn der Entwicklung des virtuellen Labors umgesetzt werden, so sollte das Konzept trotzdem von Anfang an in die *Architektur*^Artefakt integriert werden. Dadurch wird sichergestellt, dass die Mehrsprachigkeit bei einer zukünftigen Erweiterung des virtuellen Labors möglichst einfach umgesetzt werden kann (siehe dazu *Konzipiere die Einbindung von Mehrsprachigkeit*^Aktivität in Abschnitt 7.4.1.7).

7.2.3.6. Bestimme Schwierigkeitsstufen und Blickwinkel auf die Domäne

Für das virtuelle Labor ist auch zu überlegen, ob verschiedene *Schwierigkeitsstufen* (*user-level*) realisiert werden sollen. Die verschiedenen Schwierigkeitsstufen können sich beispielsweise in der Komplexität der Steuerung des virtuellen Labors, der Art der Hilfestellung und der Modellierungsgenauigkeit der Laborgeräte unterscheiden. So ist es dann zum Beispiel möglich, das virtuelle Labor als Anfänger, Fortgeschrittener oder Profi zu betreten (vergleiche Kronberg, 1998, Seite 30). Neben der Unterscheidung von verschiedenen Schwierigkeitsstufen sind auch verschiedene *Blickwinkel auf die Domäne* des virtuellen Labors denkbar. Das bedeutet, dass die zu vermittelnden Inhalte aus den Blickwinkeln verschiedener Zielgruppen dargestellt werden. Dieses Vorgehen ist besonders bei Grundlagenthemen sinnvoll, bei denen es eine gemeinsame theoretische Grundlage gibt, die aus vielerlei Richtungen betrachtet wird. Bei den verschiedenen Blickwinkeln auf die Domäne des virtuellen Labors ändern sich also die Versuchsabläufe sowie die verwendeten Substanzen und Begrifflichkeiten. Das der Domäne des virtuellen Labors zugrunde liegende Simulationsmodell bleibt jedoch gleich.

7.2.3.7. Berücksichtige die Einbindung der Lerneranalyse

Für das virtuelle Labor ist zu entscheiden, inwiefern eine Lerneranalyse durchgeführt werden soll. Die Lerneranalyse ermöglicht es dem intelligenten Tutorsystem, während der Durchführung des Versuchs die Eingaben des Lerners zu analysieren und entsprechende Hilfen zu geben. Außerdem kann eine Bewertung der Aktivitäten des Lerners vorgenommen werden. Die Konzeption der Lerneranalyse ist Gegenstand der Erstellung des *didaktischen Konzeptes*[Artefakt] und wird in der Aktivität *Konzipiere die Lerneranalyse*[Aktivität] (siehe Abschnitt 7.3.3.4) durchgeführt. Die konkrete Umsetzung der Lerneranalyse erfolgt bei der Definition der *Architektur*[Artefakt] des virtuellen Labors und wird in der Aktivität *Konzipiere die Umsetzung der Lerneranalyse*[Aktivität] (siehe Abschnitt 7.4.1.8) beschrieben.

7.2.3.8. Berücksichtige das dynamische Laden und Speichern von Versuchen

Eine weitere Anforderung an das virtuelle Labor ist das dynamische Laden von zuvor spezifizierten Versuchen. Die Versuche können dabei zum Beispiel im Lieferumfang des virtuellen Labors enthalten sein. Auch ist es denkbar, den *Endanwendern*[Rolle] neue Versuche für das virtuelle Labor im Internet zur Verfügung zu stellen. Des Weiteren sollten beliebige Zwischenzustände während der Versuchsdurchführung gespeichert und wieder geladen werden können. So kann der Lerner die Durchführung eines Versuchs an einer beliebigen Stelle unterbrechen und zu einem späteren Zeitpunkt wieder aufnehmen. Es ist auch denkbar, dass ein *Endanwender*[Rolle] einen beliebigen Zwischenzustand des virtuellen Labors an einen anderen *Endanwender*[Rolle] verschickt, zum Beispiel über E-Mail. Im Zusammenhang mit dem dynamischen Laden und Speichern von Versuchen sollte auch die Austauschbarkeit von Versuchen berücksichtigt werden. So ist es denkbar, dass Versuche im virtuellen Labor durch neuere ersetzt werden sollen, wenn sich zum Beispiel an dem Versuchsprotokoll etwas geändert hat oder wenn modernere Laborgeräte verfügbar sind. Dabei können zugleich auch neue Laborgeräte, Behälter und Substanzen in das virtuelle Labor geladen und integriert werden.

7.2.3.9. Berücksichtige die Anbindung externer Geräte (optional)

Sollen *externe Geräte*, wie zum Beispiel ein Messinstrument, an das virtuelle Labor angeschlossen werden, so sind gegebenenfalls Echtzeit-Komponenten zu entwerfen (siehe *Entwerfe die Echtzeit-Komponenten*[Aktivität] in Kapitel 7.4.5). Derartige virtuelle Labore stellen allerdings nicht den Normalfall dar. Aus diesem Grund wird die Anbindung von Echtzeit-Komponenten im *VirtLab*–Prozess als optionale Ergänzung betrachtet.

7.2.4. Betrachte den Umfang des Labors

In der Problemanalyse (siehe *Analysiere das Problem*[Aktivität] in Kapitel 7.2.1), der Definition des virtuellen Labors (siehe *Definiere das virtuelle Labor*[Aktivität] in Kapitel 7.2.3) und der

Analyse der Domäne (siehe *Analysiere die Domäne*[Aktivität] in Kapitel 7.3.2) wurde der Umfang des virtuellen Labors, das heißt die zu realisierenden Inhalte festgelegt. In dieser Aktivität wird nun der festgelegte Umfang von den *System-Analytikern*[Rolle] und dem *Projektleiter*[Rolle] betrachtet und bewertet. Wird der Umfang als zu groß eingeschätzt, so sind die Anforderungen des *Auftraggebers*[Rolle] an das virtuelle Labor zu überarbeiten. Ansonsten werden Prioritäten bezüglich der Anforderungen festgelegt und die Definition des virtuellen Labors verfeinert.

7.2.5. Verarbeite veränderte Anforderungen

In diesem Teil der *Anforderungsbestimmung*[Workflow] werden die veränderten Anforderungen an das virtuelle Labor verarbeitet. Diese Aktivität wird von den *System-Analytikern*[Rolle] durchgeführt. Veränderte Anforderungen können beispielsweise neue oder geänderte Anforderungen bezüglich der Funktionalitäten sein, oder Änderungswünsche bezüglich der Gestaltung des virtuellen Labors. Der *Projektleiter*[Rolle] entscheidet dann, welche Anforderungen berücksichtigt werden und wann die Anforderungen in die Entwicklung des virtuellen Labors einfließen. Außerdem sind gegebenenfalls neue Anwendungsfälle von den *System-Analytikern*[Rolle] zu erstellen, sowie die von den Änderungen betroffenen Anwendungsfälle zu überarbeiten.

7.3. Tutorkonzept

Der Workflow zur Erstellung des *Tutorkonzeptes*[Workflow] umfasst die didaktischen und domänenspezifischen Aspekte des virtuellen Labors und ist in Abbildung 7.10 dargestellt. Zum Aufbau der Didaktik des virtuellen Labors sind von den *Fachdidaktikern*[Rolle] neben den zu vermittelnden Lerninhalten auch die Navigationsstruktur und -möglichkeiten festzulegen. Die *Fachexperten*[Rolle] analysieren die zu modellierende Domäne und nehmen eine Aufteilung der Lerninhalte in Lerneinheiten vor. Des Weiteren stehen die *Fachexperten*[Rolle] als Ratgeber zur Verfügung und verifizieren die fachliche Korrektheit des virtuellen Labors. Außerdem führen die *Fachdidaktiker*[Rolle] die Wissenserhebung durch.

Im Workflow des *Tutorkonzeptes*[Workflow] wird zunächst die Erstellung des *didaktischen Konzeptes*[Artefakt] während der ersten Iterationen der Software-Entwicklung, also während der Iterationen der Definitions- und Entwurfsphase betrachtet. Wie aus Abbildung 7.10 zu ersehen ist, teilt sich der Workflow während der ersten Iterationen in zwei parallele Aktivitäten. Es werden die verfügbaren Quellen und Methoden zur Wissenserhebung ermittelt sowie eine detaillierte Analyse der Domäne durchgeführt. Anschließend wird das *didaktische Konzept*[Artefakt] des virtuellen Labors festgelegt. Bereits in den ersten Iterationen des *VirtLab*–Prozess ist es möglich, die Lerneinheiten und das benötigte Hintergrundwissen genau zu spezifizieren. Zudem ist in jeder Iteration eine Korrektur des *didaktischen Konzeptes*[Artefakt] möglich. Dazu sind zuvor erneut die Methoden zur Wissenserhebung zu wählen und eine Analyse der Domäne durchzuführen.

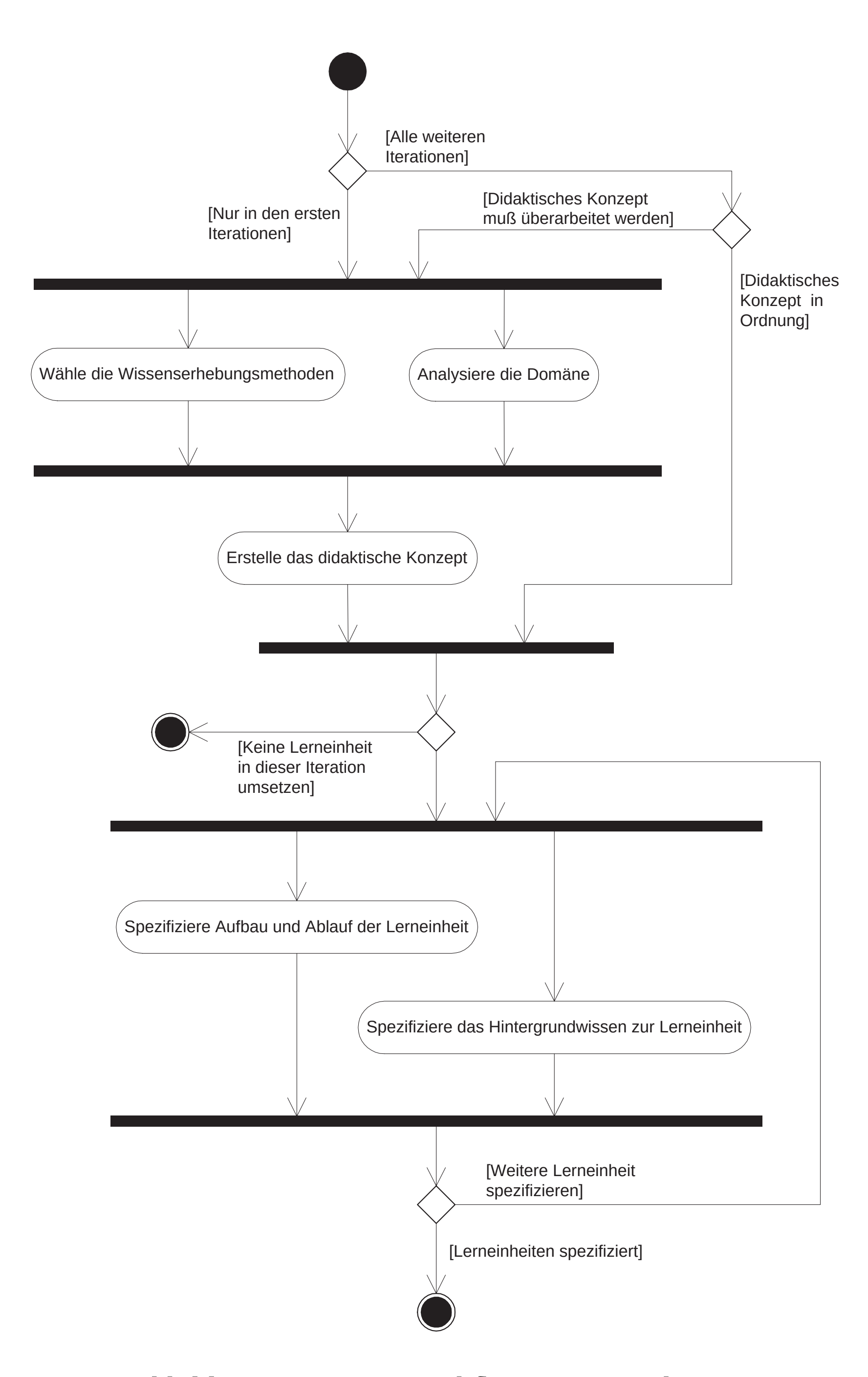


Abbildung 7.10.: Der Workflow zum Tutorkonzept

Die Aktivitäten im Workflow des *Tutorkonzeptes*^Workflow werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Wähle die Wissenserhebungsmethoden*^Aktivität (siehe Kapitel 7.3.1),
- *Analysiere die Domäne*^Aktivität (siehe Kapitel 7.3.2),

- *Erstelle das didaktische Konzept*[Aktivität] (siehe Kapitel 7.3.3),
- *Spezifiziere Aufbau und Ablauf der Lerneinheit*[Aktivität] (siehe Kapitel 7.3.4) und
- *Spezifiziere das Hintergrundwissen zur Lerneinheit* (siehe Kapitel 7.3.5).

7.3.1. Wähle die Wissenserhebungsmethoden

Für die Wahl der Methoden zur Wissenserhebung muss zunächst ermittelt werden, welche Quellen herangezogen werden sollen beziehungsweise welche Quellen zur Wissenserhebung überhaupt zur Verfügung stehen. Dabei sind die zur Verfügung stehenden Quellen von der Domäne und der Zielgruppe des virtuellen Labors abhängig. Als mögliche Quellen für die Wissenserhebung kommen prinzipiell in Frage:

- Texte: Das sind Bücher oder Dokumente der Domäne in gedruckter oder elektronischer Form, wie zum Beispiel die Versuchsprotokolle der realen Laborpraktika und das Metaobjektmodell für virtuelle Labore (siehe Kapitel 6). Aber auch schriftlich fixierte Fallbeispiele und Beobachtungen von den *Fachexperten*[Rolle] können zur Wissenserhebung verwendet werden.

- Experte: Mit Experte ist der *Fachexperte*[Rolle] aus dem Entwicklerteam des virtuellen Labors gemeint. Dieser hat umfangreiche Kenntnisse über die Domäne des virtuellen Labors und steht als Informationsquelle für die Fragen des *Fachdidaktikers*[Rolle] und der *System-Analytiker*[Rolle] zur Verfügung.

Die Texte können mittels einer Textanalyse weiterverarbeitet werden. Bei Experten können prinzipiell die Methoden des Interviews, formale Techniken und Beobachtungstechniken mit anschließender Problemanalyse durchgeführt werden. Interviews erfolgen in Form von Expertenbefragungen und werden durch Audio- und Videoaufzeichnungen unterstützt. Formale Techniken, wie zum Beispiel die Konzeptsortierung oder das Konstruktgitterverfahren (*repertory grid technique*) (siehe Schmidt, 1989; Gaines und Shaw, 1995), entsprechen im Wesentlichen dem Interview. Durch ihre formale Grundlage sind diese Techniken objektiver als Interviews, allerdings in der Regel auch weniger zielgerichtet. Eine Beobachtungstechnik mit anschließender Problemanalyse ist zum Beispiel das Ermitteln von Fallbeispielen durch die Teilnahme an einem oder mehreren realen Laborpraktika mit anschließender Auswertung der gewonnenen Informationen. Eine ausführliche Analyse und Beschreibung der Methoden zur Wissenserhebung ist zum Beispiel in (Schmidt, 1989, Seite 83 bis 117) zu finden. Die konkrete Wahl der Quellen und Methoden zur Wissenserhebung wird gemeinsam von den *Fachexperten*[Rolle] und den *Fachdidaktikern*[Rolle] unter Absprache mit dem *Projektleiter*[Rolle] und dem *Auftraggeber*[Rolle] vorgenommen. Dabei sind insbesondere die rechtlichen Aspekte zur Verwendung der Quellen, wie zum Beispiel bei den Büchern, zu berücksichtigen.

7.3.2. Analysiere die Domäne

In der *Anforderungsbestimmung*^Workflow^ wurde bereits die Zielgruppe und die Domäne des virtuellen Labors festgelegt. Parallel zur Problemanalyse (siehe *Analysiere das Problem*^Aktivität^ in Kapitel 7.2.1) wird in diesem Teil des *Tutorkonzeptes*^Workflow^ die Analyse der zu modellierenden Domäne im Detail durchgeführt (siehe Abbildung 7.11). Dazu werden die verfügbaren Quellen mit Hilfe der Methoden zur Wissenserhebung ausgewertet und aufbereitet (siehe dazu Schmidt, 1989). Es werden die *Fachexperten*^Rolle^ befragt (das heißt Interviews durchgeführt), Bücher und Versuchsprotokolle analysiert und das Metaobjektmodell für virtuelle Labore verwendet.

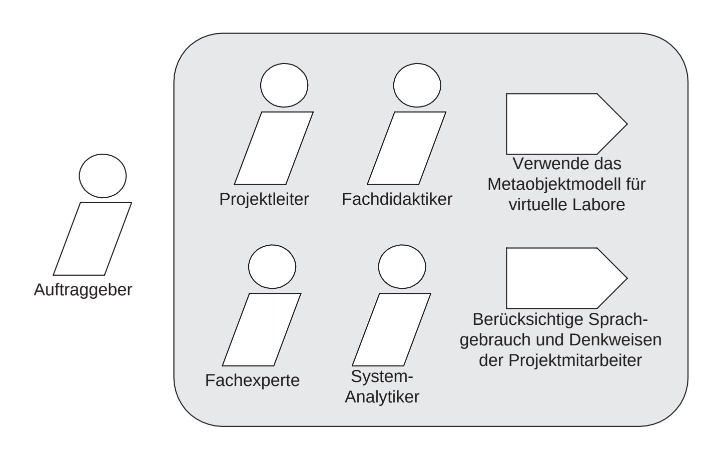


Abbildung 7.11.: Detailansicht der Aktivität Analysiere die Domäne

Die Analyse der Domäne wird gleichermaßen von den *Fachexperten*^Rolle^, *Fachdidaktikern*^Rolle^ und *System-Analytikern*^Rolle^ unter Berücksichtigung der Anforderungen des *Auftraggebers*^Rolle^ durchgeführt. Der *Projektleiter*^Rolle^ koordiniert dabei die Aktivitäten. Ergebnis dieser Aktivität ist eine Festlegung und Aufbereitung der zu vermittelnden Lerninhalte des virtuellen Labors, das heißt eine nach Prioritäten geordnete Liste von *Lerneinheiten*[3], bestehend aus den durchzuführenden Versuchen, Teilversuchen und den zu erwerbenden Fertigkeiten. Diese werden in der *Liste der Lerneinheiten*^Artefakt^ festgehalten. Wichtig ist eine möglichst optimale Reihenfolge in der Umsetzung der Versuche und insbesondere der (generischen) Teilversuche zu bestimmen, um schon während der Entwicklung des virtuellen Labors ein maximales Ergebnis in Form von durchführbaren Versuchen zu erzielen. Des Weiteren hängt die Festlegung der Prioritäten auch wesentlich von den Wünschen des *Auftraggebers*^Rolle^ ab und ist mit diesem abzusprechen. Außerdem kann zu jeder Lerneinheit die ungefähre *Lerndauer* angegeben werden. Nach der Bestimmung der *Liste der Lerneinheiten*^Artefakt^ wird gegebenenfalls die Zielgruppe des virtuellen Labors präzisiert beziehungsweise korrigiert.

[3] Auch: *Lernmodul* (siehe Ateyeh u. a., 1999, Seite 4)

7.3.2.1. Verwende das Metaobjektmodell für virtuelle Labore

Als Grundlage zur Analyse der Domäne bietet sich das in Kapitel 6 beschriebene Metaobjektmodell für virtuelle Labore an. Dazu werden die einzelnen Objekte des Metaobjektmodells betrachtet und für das konkrete virtuelle Labor identifiziert. Es werden also für jeden Versuch und jede Fertigkeit die benötigten Laborgeräte, Behälter, Substanzen und Hilfsmittel bestimmt. Außerdem wird die zugrunde liegende Theorie zu den Versuchen identifiziert und die einzuhaltenden Sicherheitsbestimmungen ermittelt.

Die Analyse der Domäne mit Hilfe des Metaobjektmodells für virtuelle Labore kann zudem durch geeignete Entwicklungswerkzeuge unterstützt werden. So ist ein Werkzeug zur Anpassung des Metaobjektmodells an die Anforderungen eines konkreten virtuellen Labors denkbar. Das Werkzeug unterstützt dabei die Definition konkreter virtueller Laborgeräte, Behälter und Substanzen sowie deren Attribute und Methoden. Ein anderes Werkzeug dient zur Unterstützung der Erstellung des Simulationsmodells. Das heißt mit dem Werkzeug kann das Zusammenspiel der Laborgeräte, Behälter und Substanzen im virtuellen Labor festgelegt werden (vergleiche die Werkzeuge *Hierarchie Definition* und *Code Composer* in Aden u. a., 1999, Seite 41).

7.3.2.2. Berücksichtige Sprachgebrauch und Denkweisen der Projektmitarbeiter

Bei der Analyse der Domäne ist es besonders wichtig, den unterschiedlichen Sprachgebrauch und die verschiedenen Denkweisen der einzelnen *Projektmitarbeiter*[Rolle] zu berücksichtigen (siehe Pasch, 1994, Seite 171). Gleiche Terminologien und Konzepte werden oft verschieden oder gar falsch verwendet und verstanden (Gaines und Shaw, 1995). Die möglichen Kombinationen der Verwendung von Terminologien und Konzepte zeigt die Abbildung 7.12.

		Terminologie	
		identisch	verschieden
Konzept	identisch	Übereinstimmung	Korrespondenz
	verschieden	Konflikt	Kontrast

Abbildung 7.12.: Verwendung von Terminologien und Konzepte

Übereinstimmungen bilden die Grundlage für die weitere Kommunikation des Entwicklerteams über gemeinsam benutzte Terminologien und Konzepte. Besonders wichtig ist es, *Konflikte* in den Konzepten zu ermitteln, um eine Verwirrung über die Benennung von verschiedenen Konzepten mit derselben Terminologie zu vermeiden. Diese Konflikte müssen ausgeräumt werden, das heißt es muss den verschiedenen Konzepten eine eindeutige

und von allen *Projektmitarbeitern*[Rolle] akzeptierte Terminologie zugeordnet werden.[4] Um eine zukünftige Verwirrung zu vermeiden, werden die Konzepte anhand der zugehörenden Terminologie in dem *Gesamtglossar*[Artefakt] schriftlich festgehalten (siehe dazu *Erstelle das (initiale) Gesamtglossar*[Aktivität] in Abschnitt 7.1.1.4). *Korrespondenz* ermöglicht ein wechselseitiges Verstehen von verschiedenen Terminologien über das Vorhandensein von gemeinsamen Konstrukten. Es ist daher sinnvoll sich auf eine Terminologie zu einigen und diese ebenfalls im *Gesamtglossar*[Artefakt] festzuhalten (gegebenenfalls unter Nennung von Synonymen).[5] In allen Dokumenten ist dieselbe Terminologie zu verwenden. Das gilt insbesondere für das *Benutzungshandbuch*[Artefakt] und die *technische Dokumentation*[Artefakt]. Das Erkennen von *Kontrasten* macht Aspekte von abweichendem Wissen deutlich. Eine gemeinsame Kommunikation und Verständnis ist dann besonders schwierig. Durch die gegebene Interdisziplinarität des Entwicklerteams virtueller Labore ist die Möglichkeit des Auftretens von Kontrasten besonders hoch. Kontraste stellen jedoch nicht unbedingt ein Problem dar, da oft dasselbe Ziel über verschiedene Wege, das heißt über verschiedene Konzepte und Terminologien, erreicht werden kann. Wichtig ist hier also die Einigung des Entwicklerteams auf ein Konzept und eine Terminologie.

Bezüglich möglicher Probleme im Zusammenhang mit dem unterschiedlichem Sprachgebrauch und den verschiedenen Denkweisen der *Projektmitarbeiter*[Rolle] sei ein Beispiel aus *GenLab* genannt. In *GenLab* hatten die *Fachexperten*[Rolle] eine andere Sichtweise auf die Domäne der Gentechnik, als die *Informatiker*[Rolle]. Während die *Fachexperten*[Rolle] die Domäne eher in den Strukturen der vorliegenden Versuchsprotokolle sehen, denken die *Informatiker*[Rolle] hauptsächlich in hierarchischen Strukturen von wiederverwendbaren Klassen und Komponenten. So wurde in *GenLab* als Grundlage der gemeinsamen Kommunikation zwischen den *Fachexperten*[Rolle] und den *Informatikern*[Rolle] ein Hierarchieschema erstellt, das die interne Struktur der Versuche beschreibt und sowohl für die *Fachexperten*[Rolle] als auch die *Informatiker*[Rolle] verständlich ist. In das Hierarchieschema konnten einerseits die *Fachexperten*[Rolle] die Versuchsprotokolle wiederfinden und andererseits konnten die *Informatiker*[Rolle] die Domäne der Gentechnik soweit verstehen, dass diese die Versuche unter Berücksichtigung wiederverwendbarer Bausteine implementieren konnten.

7.3.3. Erstelle das didaktische Konzept

Zunächst wird bei der Erstellung des *didaktischen Konzeptes*[Artefakt] (siehe Abbildung 7.13) eine Navigationsstruktur der Lerneinheiten festgelegt. Dazu wurden die Lerninhalte von den *Fachdidaktikern*[Rolle] in der Aktivität *Analysiere die Domäne*[Aktivität] (siehe Kapitel 7.3.2) in einzelne Lerneinheiten, das heißt Versuche und Fertigkeiten aufgeteilt. Die Lerneinheiten können weiter zu Gruppen geordnet werden. Eine Gruppe von Lerneinheiten ist zum Beispiel

[4] Beispiel eines Konfliktes ist die Verwendung der Terminologien Programmkontrolle (das heißt die Lenkung des Lerners durch das Lehr- und Lernsystem) und Lernerkontrolle (das heißt die Kontrolle des Lerners über das Lehr- und Lernsystem), die nach (Schulmeister, 1997, Seite 151 bis 161) oft völlig verschieden verstanden und verwendet werden.

[5] Beispiel einer solchen Korrespondenz sind die Begriffe Use-Case und Anwendungsfall, als dessen Übersetzung ins Deutsche.

eine Menge von zusammenhängenden Versuchen oder Fertigkeiten. Zu jeder Lerneinheit wird außerdem ein festes Lernziel definiert, das mit dem erfolgreichen Abschluss der Lerneinheit erreicht wird (vergleiche Issing, 1997, Seite 203). Bei einem Versuch wird das Lernziel mit der erfolgreichen Durchführung aller Arbeitsschritte des entsprechenden Versuchsprotokolls erreicht. Für eine Fertigkeit werden ähnlich wie in einem Versuchsprotokoll, eine bestimmte Anzahl an Arbeitsschritten festgelegt. Diese sind jedoch wesentlich feingranularer als bei den Versuchen und mit entsprechenden Hintergrundinformationen versehen. Das Lernziel einer Fertigkeit wird analog zu den Versuchen mit der erfolgreichen Durchführung der entsprechenden Arbeitsschritte definiert.

Bezüglich der Navigationsmöglichkeiten in den Lerneinheiten ist auch das *Interaktionsdesign* des virtuellen Labors zu berücksichtigen. Als *Interaktionsformen* stehen dazu hauptsächlich graphische Bedienelemente (*User-Interface Komponenten*) und direktmanipulative Metaphern (Ziegler, 1993, Seite 146 ff.) zur Verfügung (Boles, 1994, Seite 41 bis 45). Es ist zum Beispiel zu entscheiden, ob sich die Substanzen zu einem Versuch in den entsprechenden Behältern auf der Benutzungsoberfläche des virtuellen Labors befinden sollen, oder ob das graphische Bedienelement der *Dialogbox* zur Auswahl der Substanzen angeboten wird (siehe Hasler und Schlattmann, 2001, Seite 20 und 21). Die Dialogbox muss dann bei der Erstellung der Benutzungsoberfläche entsprechend gestaltet und die dafür benötigten Medien in der *Medienproduktion*[Workflow] hergestellt werden. Auch ist zum Beispiel zu überlegen, ob die Tür der Mikrowelle geöffnet werden muss, um einen Behälter hineinzustellen, oder ob die direktmanipulative Metapher des *Drag & Drop*-Mechanismus (siehe *Konzipiere Mechanismen der explorativen Lehr- und Lernumgebung*[Aktivität] in Abschnitt 7.4.1.4) dazu ausreicht, da vom Lerner die Fähigkeit zur Bedienung der Mikrowelle vorausgesetzt werden kann. Außerdem ist zu bestimmen welche *Eingabegeräte*, wie zum Beispiel Computer-Maus und Tastatur, in welcher Weise für die Interaktionsformen verwendet werden können (siehe dazu Boles, 1994, Seite 39 ff. und 45 ff.). Abschließend bleibt zu überlegen, ob die experimentbezogene und die informationsbezogene Sicht auf das Metaobjektmodell auf der Ebene der Benutzungsoberfläche getrennt oder miteinander verknüpft werden sollen (siehe dazu den Zusammenhang der beiden Sichten in Kapitel 6.3).

7.3.3.1. Bestimme die Navigationsstruktur über die Lerneinheiten

Durch die Bestimmung der Lerneinheiten und Lernziele lassen sich logische Abhängigkeiten zwischen den einzelnen Lerneinheiten ermitteln. Eine logische Abhängigkeit zwischen den Lerneinheiten ist zum Beispiel, dass für das Verständnis der Lerneinheit A der Inhalt der Lerneinheit B vorausgesetzt wird (Ateyeh u. a., 1999, Seite 9). Für die Navigationsstruktur im virtuellen Labor bedeutet das konkret, dass die Lerneinheit A erst dann von dem Lerner ausgewählt werden kann, wenn die Lerneinheit B erfolgreich beendet worden ist. Diese Abhängigkeiten ergeben eine *Hierarchie der Lerneinheiten* und stellen zugleich das logische Grundgerüst des *didaktischen Konzeptes*[Artefakt] dar (*Aufgabenkaskade*). Diese Hierarchie der Lerneinheiten ist die kanonische Festlegung einer logisch geordneten *Navigationsstruktur über die Lerneinheiten* und bestimmt, welche Lerneinheiten wann ausgewählt werden können oder gesperrt sind (vergleiche den logischen Abhängigkeitsgraphen in Ateyeh u. a., 1999,

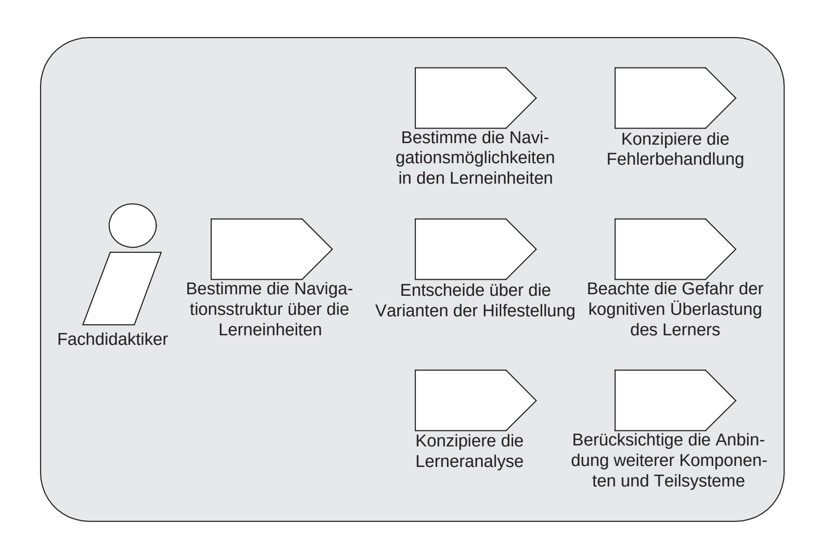


Abbildung 7.13.: Detailansicht der Aktivität Erstelle das didaktische Konzept

Seite 9). Mit einer *Guided Tour* wird dem Lerner ein bestimmter Ablauf durch die Hierarchie der Lerneinheiten des virtuellen Labors vorgeschlagen. Es ist auch möglich, anhand des Lernermodells zu entscheiden, welche Lerneinheiten ausgewählt werden können. So besitzen erfahrene Lerner die grundlegenden Fertigkeiten bereits und können diese überspringen. Eine weitere Möglichkeit ist, dass immer alle Lerneinheiten auswählbar sind (*freie Navigation*). Das ist dann notwendig, wenn sich keine Hierarchie der Lerneinheiten bilden lässt oder kein Lernermodell existiert und somit der Wissensstand des Lerners nicht bekannt ist. Die freie Navigation kann aber auch bewusst eingesetzt werden, um dem Lerner eine größtmögliche Auswahl an Lerneinheiten anzubieten, wie zum Beispiel in *GenLab*. Des Weiteren ist die freie Navigation eine wesentliche Anforderung an erwachsenengerechte Lehr- und Lernsysteme (siehe Schanda, 1995, Seite 176). So sieht sich der erwachsene Lerner denn auch eher als aktiver, selbstgesteuerter Lerner denn als bloßer Rezipient von Information (Gallenberger u. a., 1999, Seite 259). Weitere denkbare Navigationsstrukturen über die Lerneinheiten sind die Suche nach Stichworten, Suchbegriffen oder über das Glossar (vergleiche Navigationsstrukturen auf Lerneinheiten in Ateyeh u. a., 1999, Seite 14).

7.3.3.2. Bestimme die Navigationsmöglichkeiten in den Lerneinheiten

Für die *Navigationsmöglichkeiten innerhalb der Lerneinheiten* stehen für virtuelle Labore die verschiedensten *Steuerungskonzepte* zur Verfügung. Diese Steuerungskonzepte unterscheiden sich hinsichtlich der Freiheit der Benutzerinteraktion. Diesbezüglich sind folgende Varianten denkbar (vergleiche Hasler und Schlattmann, 2001, Seite 11 f.):

- *Freie Versuchsmöglichkeit*: Das virtuelle Labor agiert als explorative Lehr- und Lernumgebung und beschränkt sich im Wesentlichen auf die Funktionalitäten des Simulators. Alle Arbeitsplätze können vom Lerner betreten und sämtliche Laborgeräte, Behälter und Substanzen nach belieben verwendet werden. Dabei kann der Lerner

entweder zuvor einen speziellen Versuch auswählen, oder der Lerner betritt das virtuelle Labor ohne einen bestimmten Versuch ausgewählt zu haben (vergleiche Elfreich, 1999, Seite 24). Im letzteren Fall kann der Lerner das virtuelle Labor zum freien Erforschen der Reaktionsabläufe verwenden (experimentieren).

- *Sperren von Arbeitsplätzen*: Im virtuellen Labor sind nur bestimmte Arbeitsplätze auswählbar. Da Arbeitsplätze oft einen inhaltlichen Bezug haben (siehe *Bestimme Anzahl und Ausstattung der Laborräume*^{Aktivität} in Abschnitt 7.2.3.2), bedeutet das Sperren von Arbeitsplätzen auch, dass bestimmte Versuche nicht durchgeführt werden können.

- *Sperren von Laborgeräten und Behältern*: Werden bestimmte Laborgeräte oder Behälter auf einem oder mehreren Arbeitsplätzen nicht benötigt oder wurde der Lerner im Verlauf der Versuchsdurchführungen noch nicht mit den Laborgeräten vertraut gemacht, so ist es sinnvoll diese zu sperren. Es ist aber auch möglich, die Laborgeräte und Behälter erst dann in der virtuellen Laborumgebung darzustellen, wenn diese auch tatsächlich benötigt werden. So wird der Lerner nicht durch gesperrte, das heißt nicht benötigte oder verwendbare Gegenstände im virtuellen Labor verwirrt und kann sich vollständig auf den durchzuführenden Versuch und die dafür benötigten Versuchskomponenten konzentrieren.

- *Sperren einzelner Knöpfe, Schalter und Regler von Laborgeräten*: Das Sperren einzelner Knöpfe, Schalter und Regler von Laborgeräten oder sonstigem Zubehör ist dann sinnvoll, wenn der Lerner einzelne Fertigkeiten erlernen soll, wie zum Beispiel die Bedienung eines bestimmten Laborgerätes oder eine typische Handlungsabfolge. Der Lerner erfährt dabei über das intelligente Tutorsystem, welche Aktivitäten zur erfolgreichen Bearbeitung der Lerneinheit durchzuführen sind.

- *Geführter Versuchsablauf*: Bei dem geführten Versuchsablauf ist immer nur ein Knopf, Schalter oder Regler zur selben Zeit während der Durchführung eines Versuchs aktiviert. Alle anderen Knöpfe, Schalter und Regler und insbesondere auch der Arbeitsflächenwechsel sind gesperrt (siehe oben). Der geführte Versuchsablauf wurde zum Beispiel bei dem *Virtuellen Sequenzierlabor* (ViSeL) der Universität Bielefeld realisiert (siehe ViSeL - Internetseiten, 2001).

- *Keine Interaktionsmöglichkeit*: Der Versuchsablauf wird dem Lerner in Form einer Animation mit Hilfe des Simulators vorgeführt, ohne das für den Lerner irgendeine Möglichkeit besteht, in den Versuch einzugreifen. Die einzigen aktiven Interaktionselemente der Benutzungsoberfläche sind die Knöpfe zum Starten, Stoppen, Unterbrechen und Beenden der Animation.

Des Weiteren sind auch Mischformen der Navigationsmöglichkeiten in den Lerneinheiten denkbar. So kann beispielsweise die freie Versuchsmöglichkeit, bei der der Lerner zuvor einen speziellen Versuch ausgewählt hat, mit dem Sperren von Arbeitsflächen, die für den ausgewählten Versuch nicht benötigt werden, kombiniert werden.

7.3.3.3. Entscheide über die Varianten der Hilfestellung

Neben den Navigationsmöglichkeiten in den Lerneinheiten, das heißt die Freiheit der Benutzerinteraktionen (siehe Abschnitt 7.3.3.2), unterscheiden sich die Steuerungskonzepte für virtuelle Labore auch in der *Qualität der Hilfestellung*. Dabei sind folgende Varianten denkbar:

- *Keine Hilfestellung*: Das intelligente Tutorsystem gibt dem Lerner während der Abarbeitung einer Lerneinheit keinerlei Hilfestellung. Diese Variante ist beim freien Experimentieren sinnvoll.

- *Ständige Hilfestellung*: In jedem Schritt der Lerneinheit wird von dem intelligenten Tutorsystem eine Hilfestellung eingeblendet. Dabei kann der Umfang der dargestellten Hilfe zwischen einer ausführlichen Beschreibung des nächsten Arbeitsschrittes und einer sehr knapp gehaltenen Übersicht über den kompletten Versuchsablauf (*Kurzanleitung*) variieren. Die Kurzanleitung stellt also eine sehr stark gekürzte Version eines Versuchsprotokolls dar. Die Variante der ständigen Hilfestellung eignet sich besonders gut für den Erwerb von Fertigkeiten und Handlungsabläufen.

- *Hilfe auf Anfrage*: Der Lerner erhält auf eigenen Wunsch eine Hilfestellung vom intelligenten Tutorsystem. Es entscheidet also der Lerner, wann und wie oft eine Hilfestellung gegeben wird (im Gegensatz zur adaptiven Hilfe).

- *Adaptive Hilfe*: Bei der adaptiven Hilfe entscheidet das Lehrermodul des intelligenten Tutorsystems anhand des Lernermodells, wann und wie oft der Benutzer eine Hilfestellung erhält, das heißt der Lerner unterbrochen wird (vergleiche Mikro-Adaption in Leutner, 1997, Seite 143 f.). Diese Art der Hilfestellung ist bei der Durchführung von Versuchen zum Erwerb von Handlungswissen und Wissen über die Reaktionsabläufe im Versuch sinnvoll. Typische Situationen, in der das intelligente Tutorsystem den Lerner unterbricht, sind zum Beispiel häufig gemachte Fehler und Stocksituationen (siehe dazu *Konzipiere die Fehlerbehandlung*^{Aktivität} in Abschnitt 7.3.3.5).

Zwischen den genannten Möglichkeiten der Hilfestellung sind eine Reihe von Abstufungen möglich. Auch schließen sich die Varianten nicht unbedingt gegenseitig aus. So ist zum Beispiel eine Kombination von ständiger Hilfestellung und adaptive Hilfe denkbar und wurde zum Beispiel in Form eines Anleitungsfensters (siehe Kapitel 2.8) in dem Projekt *GenLab* realisiert (siehe Hasler und Schlattmann, 2001, Seite 12). Das Anleitungsfenster enthält eine Kurzanleitung zum Versuchsprotkoll und kann vom Lerner während der Versuchsdurchführung beliebig oft eingesehen werden. In einer Stocksituation kann sich der Lerner den nächsten Arbeitsschritt in Form einer Animation vorführen lassen. Ist ein Arbeitsschritt bereits bekannt, so kann dieser vom Lerner übersprungen werden. Außerdem wird der Lerner auf Fehler bei der Versuchsdurchführung aufmerksam gemacht und muss die entsprechenden Arbeitsschritte wiederholen. Dieses Beispiel einer einfachen adaptiven Hilfe in Verbindung mit einer freien Experimentierumgebung sieht ein Lernermodell vor, das im Wesentlichen aus der Kurzanleitung des Versuchsprotokolls besteht. Das Lernermodul

ermittelt den Wissensstand des Lerners, indem es den aktuellen Zustand des virtuellen Labors beobachtet. Das Lehrermodul vergleicht die Beobachtungen mit der Kurzanleitung im Anleitungsfenster und nimmt entsprechende Eintragungen vor. Des Weiteren ist für eine umfassende adaptive Hilfe auch der Einsatz eines Expertensystems denkbar (siehe *Entwerfe das Expertensystem*^{Aktivität} in Kapitel 7.4.7).

Unabhängig von den genannten Varianten können die Inhalte der Hilfestellung in *inhaltsbezogene* und *interaktionstechnische Hilfestellung* unterschieden werden. Die inhaltsbezogene Hilfestellung betrifft die Domäne des virtuellen Labors und liefert zum Beispiel Information zu einem Arbeitsschritt des Versuchsablaufs oder zu einer Versuchskomponente. Bei der interkationstechnischen Hilfestellung handelt es sich um Informationen bezüglich der Bedienung des virtuellen Labors und insbesondere der Laborgeräte und Behälter sowie der Labornavigation.

7.3.3.4. Konzipiere die Lerneranalyse

Die *Lerneranalyse* ist sowohl Bestandteil des Lerner- als auch des Lehrermoduls. Für die Konzeption der Lerneranalyse sind eine Reihe von Aspekten zu berücksichtigen (siehe auch Diagnose des Wissenstandes des Lernenden in Witschital, 1990, Seite 47 bis 50). So muss untersucht werden, welche Kriterien den Lernerfolg ausmachen, das heißt nach welchen Gesichtspunkten der Lernerfolg gemessen werden soll. Des Weiteren ist zu entscheiden, welche Hilfsmittel neben dem virtuellen Labor noch als Informationsquelle erlaubt sind. Um eine objektive Bewertung zu garantieren, kann zum Beispiel sichergestellt werden, dass das virtuelle Labor die einzige Informationsquelle des Lerners ist. Für die Umsetzung der Lerneranalyse sind Daten des Lerners (*Lernerdaten*) zu protokollieren (siehe *Software-Recording* in Freibichler, 2000, Seite 311 f.). Unter Lernerdaten werden sämtliche Informationen verstanden, die vom Lernermodul aus den Eingaben des Lerners über Computer-Maus und Tastatur ermittelt werden können (Witschital, 1990, Seite 47). Wichtig bei der Protokollierung von Lernerdaten ist die Diskriminierung, das heißt Unterscheidung relevanter Information. Um die relevanten Daten gezielt protokollieren zu können, ist es sinnvoll, zuvor eine geeignete *Protokoll-Struktur* zu erstellen (Förster und Zwernemann, 1993, Seite 75). So ist es zum Beispiel eher unerheblich zu protokollieren, wie oft beziehungsweise wie lange einzelne Versuchskomponenten mit der Computer-Maus bewegt wurden. Auch ist es gegebenenfalls vernachlässigbar, ob ein Reaktionsgefäß mit der *Hand*, das heißt der Computer-Maus, oder mit einem entsprechenden Laborgerät, wie zum Beispiel dem Vortexer, geschüttelt wurde. Wichtig ist dagegen zum Beispiel die Protokollierung der Reihenfolge, in der die einzelnen Arbeitsschritte eines Versuchs durchgeführt werden und die zur Durchführung der einzelnen Arbeitsschritte benötigte Zeit. So können prinzipiell alle Arbeitsschritte streng sequenziell nach den Vorgaben des Versuchsprotokolls abgearbeitet werden. Unter Umständen erspart es jedoch Zeit- und Materialkosten, wenn bestimmte Arbeitsschritte parallel oder in einer anderen Reihenfolge durchgeführt werden. Auch sind die Anzahl und Art der gemachten Fehler zu protokollieren, da diese wichtige Informationen über den Lernerfolg liefern.

Um den Lernerfolg objektiv zu messen, sind die einzelnen Aktionen des Lerners auszuwerten und zu bewerten. Dafür wird basierend auf der Protokoll-Struktur eine *Punktebewertung* eingeführt. Für jede Aktivität des Lerners wird eine Punktezahl vergeben. Im einfachsten Fall ergibt sich die Punktebewertung der Lerneinheit durch einfaches Aufsummieren oder durch bilden des Durchschnitts über die Punktezahlen der einzelnen Aktionen des Lerners während der Durchführung der Lerneinheit (Russell und Norvig, 1995, Seite 47 f.). Zu bewerten ist zum Beispiel die Reihenfolge, in der die einzelnen Arbeitsschritte eines Versuches durchgeführt wurden, die dafür benötigte Zeit sowie die Anzahl und Schwere der Fehler. Aber auch die verstrichene Zeit, die der Lerner für die Abarbeitung eines Versuches benötigt, fließt in die Punktebewertung mit ein. Die Punktebewertung gibt Auskunft darüber, wie erfolgreich ein Lerner bei der Abarbeitung einer Lerneinheit war (vergleiche *performance measure* in Russell und Norvig, 1995, Seite 32). Sie dient dabei vornehmlich zur Motivation der Lerners (*kompetitives Lernen*) sowie zur Unterstützung der Nachanalyse der Lernerdaten (siehe *Analysiere Ergebnisse der Evaluation*^Aktivität in Kapitel 7.7.8). Die Punktebewertung kann aber prinzipiell auch zur Notenvergabe im Rahmen einer schulischen oder universitären Veranstaltung verwendet werden.

Das Protokollieren von Daten des Lerners ist nach (Förster und Zwernemann, 1993, Seite 75) ein umstrittenes Thema. Demnach sollte der Grundsatz gelten, dass so viele Daten wie nötig und so wenige wie möglich protokolliert werden. Werden Lernerdaten protokolliert, so ist der Lerner darüber zu informieren. Des Weiteren ist der Erhebungszweck dem Lerner gegenüber anzugeben (siehe Bundesdatenschutzgesetz, 2001, §§13 und 14).

7.3.3.5. Konzipiere die Fehlerbehandlung

Ein weiterer wichtiger Aspekt des *didaktischen Konzepts*^Artefakt ist die *Behandlung von Fehlern*. Diese ist wie die Lerneranalyse ebenfalls Bestandteil des Lerner- aber auch des Lehrermodells. Im Lernermodell muss festgelegt werden, wie die Fehler des Lerners ermittelt und welche Typen von Fehlern erkannt werden sollen. Teil des Lehrermodells ist es, diese Fehler zu analysieren und darauf unter Verwendung der Informationen aus dem Expertenmodell entsprechend zu reagieren (vergleiche Witschital, 1990, Seite 31). Durch Beobachtungen von realen Laborpraktika und Befragungen der *Fachexperten*^Rolle können häufig gemachte Fehler ermittelt und anschließend analysiert werden. Die daraus gewonnenen Erkenntnisse fließen in das Lernermodell als auch das Lehrermodell ein. Allgemein sollte das intelligente Tutorsystem nur dann eingreifen, wenn der Lerner einen Fehler macht oder der Eingriff als didaktisch sinnvoll erscheint, zum Beispiel um wichtiges Hintergrundwissen zu den gerade ablaufenden Reaktionsprozessen darzustellen (vergleiche Witschital, 1990, Seite 54). Wird ein Flüchtigkeitsfehler gemacht, reicht eine Warnung oder ein Hinweis des intelligenten Tutorsystems über die Benutzungsoberfläche aus. Bei bekannten beziehungsweise häufig gemachten Fehlern (*pitfall*) bietet sich eine detailierte Erklärung der Fehlerursache an. Werden schwerwiegende Fehler gemacht, zum Beispiel bei Verständnisproblemen, so sollte das intelligente Tutorsystem den Lerner die Lerneinheit wiederholen lassen und zusätzliche Hilfen geben. Das intelligente Tutorsystem sollte den Lerner ebenfalls unterbrechen, wenn aufgrund des Lernermodells, das heißt den

Beobachtungen des Lernermoduls[6], auf eine Stocksituation des Lerners geschlossen werden kann.

7.3.3.6. Beachte die Gefahr der kognitiven Überlastung des Lerners

Bei der Erstellung des *didaktischen Konzeptes*[Artefakt] ist des Weiteren auf eine mögliche *kognitive Überlastung* (*cognitive overload*) des Lerners zu achten. Unter der kognitiven Überlastung des Lerners versteht man die Tatsache, dass aufgrund der Vielzahl von Handlungsalternativen in einer gegebenen Situation, wie zum Beispiel das Angebot mehrerer Handlungsalternativen anstelle einer klar bestimmten Arbeitsabfolge, ein Teil der Aufmerksamkeit des Lerners dafür aufgewendet werden muss. Das kann dazu führen, dass sich der Lerner überwiegend mit der Handhabung der Benutzungsoberfläche beschäftigen muss und nicht mit den Lerninhalten (siehe Blumstengel, 1998, Kapitel 4). Bei virtuellen Laboren kann die Gefahr der kognitiven Überlastung des Lerners zum Beispiel dadurch vermieden werden, dass sich nur die für den Versuch notwendigen Versuchskomponenten auf den Arbeitsflächen befinden und der Lerner die Möglichkeit hat, die Versuche anhand eines im Anleitungsfenster dargestellten Versuchsprotokolls durchzuführen.

7.3.3.7. Berücksichtige die Anbindung weiterer Komponenten und Teilsysteme

Sollen weitere Komponenten und Teilsysteme an das virtuelle Labor angebunden werden, wie zum Beispiel ein klassisches CBT-System (*Computer Based Training*), das heißt ein Lehr- und Lernsystem im Sinne von Kapitel 2.4, so ist dieses zu konzipieren und eine Schnittstelle zum virtuellen Labor festzulegen. Für die Entwicklung von klassischen Lehr- und Lernsystemen sei auf die zahlreiche Literatur zu diesem Thema verwiesen. So können zum Beispiel die in Kapitel 3.3.2 und 3.3.3 beschriebenen Vorgehensmodelle nach (Yass, 2000) und (Nagl u. a., 1999) für die Entwicklung von klassischen Lehr- und Lernsystemen verwendet werden.

Aber auch die Anbindung eines Online-Handbuchs (siehe *Erstelle das Benutzungshandbuch*[Aktivität] in Abschnitt 7.8.2.1) an das virtuelle Labor ist denkbar. So wurde in *GenLab* das Anleitungsfenster der Laborkomponente über Querverweise mit der Informationskomponente verbunden (siehe Kapitel 6.3). Zur Erstellung einer Informationskomponente wie in *GenLab* kann zum Beispiel das in (Sawhney, 1995) vorgestellte Vorgehensmodell für die Multimedia-Anwendungsentwicklung verwendet werden (siehe Kapitel 3.3.1).

Eine weitere Ergänzung zum virtuellen Labor ist die Integration eines *Notizbuchs*. In dieses kann der Lerner während der Durchführung der Versuche und dem Erwerb der Fertigkeiten Informationen in textueller und graphischer Form nach belieben eintragen und wieder abrufen.

[6] Zum Beispiel: Auswertung der Bewegungsdaten des Mauszeigers

7.3.4. Spezifiziere Aufbau und Ablauf der Lerneinheit

In diesem Teil des *Tutorkonzeptes*[Workflow] erfolgt die konkrete Spezifikation von Aufbau und Ablauf der Lerneinheiten, die in der aktuellen Iteration umgesetzt werden sollen (siehe Abbildung 7.14). Eine Lerneinheit stellt im virtuellen Labor die Durchführung eines Versuchs oder den Erwerb einer Fertigkeit dar (siehe Kapitel 6.1). Die Spezifikation der Lerneinheit erfolgt mit Hilfe einer speziellen, von der Domäne des virtuellen Labors unabhängigen Spezifikationssprache.

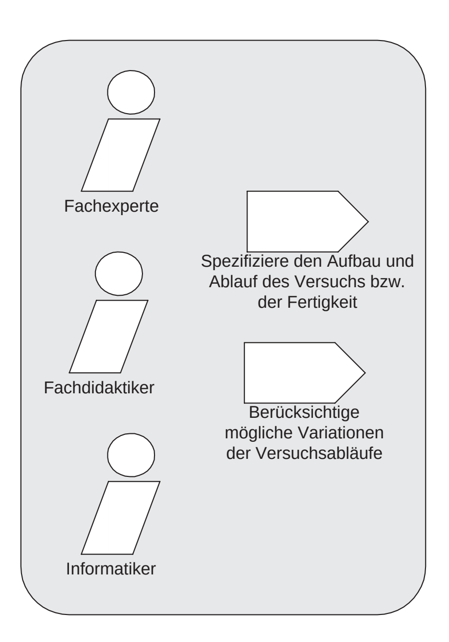


Abbildung 7.14.: Detailansicht der Aktivität Spezifiziere Aufbau und Ablauf der Lerneinheit

Allgemein ist bei der Spezifikation der Lerneinheit zu entscheiden, welches Steuerungskonzept verwendet werden soll beziehungsweise welches das geeigneteste ist. Eine gute Hilfe stellen dabei die Versuchsprotokolle dar. Die Spezifikation des Aufbaus und Ablaufs der Lerneinheit kann in einer formalen, halb-formalen oder nicht-formalen Spezifikationssprache erfolgen. Damit verbunden ist die Modellierungsebene der Spezifikationssprache. Die konkrete Durchführung einer Spezifikation lässt sich außerdem noch darin unterscheiden, ob ein entsprechendes Entwicklungswerkzeug zur Verfügung steht oder nicht.

7.3.4.1. Spezifiziere Aufbau und Ablauf des Versuchs bzw. der Fertigkeit

Für die Spezifikationssprache sind prinzipiell zwei Modellierungsebenen denkbar. Zum einen kann die Spezifikation der Lerneinheiten auf der *Wissensebene* und zum anderen auf der *Symbolebene* erfolgen (vergleiche Reimer, 1991). Eine Spezifikation auf Wissensebene beschreibt, was in dem Versuch zu tun ist. Wie die konkrete Durchführung des Versuchs aussieht, bleibt auf der Wissensebene verborgen, das heißt die internen Strukturen der Versuchsspezifikation werden automatisch von dem virtuellen Labor um entsprechende Funktionsaufrufe ergänzt. Die Modellierung auf Wissensebene erfolgt mit Hilfe einer

graphischen Notation, die allgemein verständlich ist (insbesondere für die *Fachexperten*^{Rolle}). Im Gegensatz zur Wissenebene enthält eine Spezifikationssprache auf Symbolebene weitere Sprachelemente zur Beschreibung der internen Strukturen. Eine Spezifikation auf Symbolebene erfolgt über eine spezielle Programmiersprache, wie zum Beispiel in *GenLab*. Zwischen den Spezifikationssprachen auf Wissensebene und Symbolebene existiert ein fließender Übergang. So kann eine auf die Wissensebene ausgerichtete Spezifikationssprache auch einige Elemente zur Beschreibung interner Strukturen enthalten.

Spezifikation von Aufbau und Ablauf mit Entwicklungswerkzeug: Die Spezifikation der Lerneinheiten mit Hilfe eines entsprechenden Entwicklungswerkzeuges basiert auf einer formalen Spezifikationssprache. Das Werkzeug ermöglicht es den *Fachexperten*^{Rolle} die Versuche und Fertigkeiten mittels einer graphischen Benutzungsoberfläche zu erstellen, ohne dabei vertiefte Kenntnisse in der Informatik haben zu müssen. Mit Hilfe des Werkzeuges können Aufbau und Ablauf der Lerneinheit festgelegt werden. Treten Probleme mit der Bedienung des Werkzeuges auf oder reicht das Werkzeug nicht für die Modellierung eines bestimmten Versuches aus, so werden die *Informatiker*^{Rolle} zu Rate gezogen. Im Idealfall erfolgt die Spezifikation der Versuche bei Verwendung eines entsprechenden Entwicklungswerkzeuges auf der Wissensebene. Die verwendete Spezifikationssprache abstrahiert dann soweit von den internen Strukturen eines Versuches, dass das Werkzeug direkt, das heißt ohne aufwendige Einarbeitung in die Spezifikationssprache, von den *Fachexperten*^{Rolle} angewendet werden kann. Ebenso ist aber auch ein Werkzeug zur Spezifikation der Versuche und Fertigkeiten auf Symbolebene denkbar. Die entsprechende Spezifikationssprache enthält dann Sprachelemente zur Beschreibung der internen Strukturen. Dies hat zur Folge, dass die Spezifikationssprache von den *Fachexperten*^{Rolle} nicht mehr intuitiv verstanden werden kann. Unabhängig von der Modellierungsebene können die Versuchsabläufe, die mit einem auf einer formalen Spezifikationssprache basierenden Werkzeug erstellt wurden, direkt von dem virtuellen Labor interpretiert und dargestellt werden. Oder die formale Spezifikation wird mittels einer automatisierten Generierung in einen ausführbaren Versuch des virtuellen Labors umgesetzt. Beispiel einer formalen Spezifikationssprache ist die *Experiment Description Language* (EDL) für virtuelle Labore (siehe Aden u. a., 1999, Seite 104 bis 118). Damit lassen sich beliebige Versuche und Fertigkeiten auf Symbolebene spezifizieren. Die Spezifikation wird dann zur Laufzeit vom virtuellen Labor interpretiert und ausgeführt. Derzeit wird im Rahmen von *VirtLab* außerdem die *Extensible Markup Language*[7] (XML) zur Beschreibung von Aufbau und Ablauf von Versuchen erprobt.

Spezifikation von Aufbau und Ablauf ohne Entwicklungswerkzeug: Steht für die Spezifikation der Versuche und Fertigkeiten im virtuellen Labor kein geeignetes Werkzeug zu Verfügung, so sind Aufbau und Ablauf der Lerneinheiten direkt zu implementieren. Reicht das Werkzeug zur Spezifikation der Versuchsabläufe nicht aus, weil zum Beispiel Laborgeräte, Zubehör oder Substanzen im virtuellen Labor fehlen, so ist das Werkzeug

[7] Siehe http://www.w3.org/XML/

entsprechend zu erweitern und die fehlenden Versuchskomponenten zu implementieren. In dem hier betrachteten Fall, dass kein Entwicklungswerkzeug zur Verfügung steht, wird davon ausgegangen, dass für die durchzuführenden Versuche des virtuellen Labors entsprechende *Versuchsprotokolle* aus einem realen Laborpraktikum vorliegen. Versuchsprotokolle stellen eine halb-formale Beschreibung eines Versuchsablaufs dar. Sie enthalten eine knappe und in der Regel sequenzielle Beschreibung der Arbeitsschritte, die für die Durchführung eines Versuches notwendig sind und werden von den *Fachexperten*^{Rolle} erstellt. Bei der Spezifikation der Lerneinheiten ist allerdings zu berücksichtigen, dass die Arbeitsschritte zwar sequenziell im Versuchsprotokoll beschrieben werden, es aber prinzipiell möglich ist, dass Arbeitsschritte auch in einer anderen Reihenfolge abgearbeitet werden können. Besonders problematisch bei den Versuchsprotokollen ist aber, dass oft wichtige Zusatzinformationen nicht enthalten sind und implizit als bekannt vorausgesetzt werden. Dazu ein Beispiel aus *GenLab* (siehe auch Anhang A):

> „$1g$ Agarose wird in $100ml$ TBE-Puffer aufgekocht, auf $65°C$ abgekühlt und in gelöster Form nach Zugabe von Ethidiumbromid (Endkonzentration $1\mu g/ml$) in den Gelträger gegossen."

Für einen *Fachexperten*^{Rolle} reicht diese Beschreibung völlig aus. Damit aber auch *Projektmitarbeiter*^{Rolle}, die keine *Fachexperten*^{Rolle} sind, die Versuchsprotokolle verstehen können, müssen diese um folgende Informationen ergänzt werden:

- *Verwendete Laborgeräte, Behälter und Hilfsmittel*: Es müssen die verwendeten Laborgeräte aufgelistet werden, sowie die benötigten Behälter und Hilfsmittel. So fehlt im obigen Beispiel die Information, dass die Agarose mit einer Waage abgewogen wird, das TBE-Puffer mit einem Messbecher oder Messzylinder abzumessen ist und das Gemisch letztlich in der Mikrowelle aufgekocht wird. Bezüglich der Behälter ist des Weiteren anzugeben, welches Volumen dieser aufnehmen kann.

- *Bedienung der Laborgeräte, Behälter und Hilfsmittel*: Neben der Auflistung der verwendeten Laborgeräte, Behälter und Hilfsmittel muss auch deren Bedienung erklärt werden. Es muss also beschrieben werden, wie die Waage und die Mikrowelle zu bedienen sind und wie mit einem Messbecher eine Substanz abgemessen werden kann. Des Weiteren müssen zu den Laborgeräten genaue Angaben bezüglich der möglichen Einstellungen der Schalter und Knöpfe gemacht werden. Auch haben die Anzeigen und Messskalen der verwendeten Laborgeräte und Behälter, die zur Abmessung von Substanzen verwendet werden, bestimmte Ungenauigkeiten, die bekannt sein müssen.

- *Erlaubte Abweichungen bei den Mengenangaben*: In den Versuchsprotokollen werden die Mengen der benötigten Substanzen oft exakt angegeben, wie zum Beispiel $1g$ Agarose oder $100ml$ TBE-Puffer. Für die Erstellung des Simulationsmodells muss bekannt sein, wie groß die Abweichungen zu den exakten Mengenangaben sein dürfen, so dass trotzdem eine erfolgreiche Versuchsdurchführung sichergestellt ist. Des Weiteren sollte für das Simulationsmodell bekannt sein, was passiert, wenn die Toleranzgrenzen der Mengenangaben über- beziehungsweise unterschritten werden und wenn falsche Substanzen verwendet werden.

- *Ungenaue beziehungsweise funktionale Mengenangaben:* Die Versuchsprotokolle können aber auch nur ungenaue Mengenangaben der benötigten Substanzen enthalten, wie zum Beispiel die Mengenangabe einen kleinen Spatel oder einen großen Spatel der Substanz zu nehmen (dann: *funktionale Mengenangabe*). Für das Simulationsmodell muss bekannt sein, wie viel Gramm der entsprechenden Substanz ein Spatel bestimmter Größe im Durchschnitt enthält und wie groß die Abweichungen sein können.

- *Beschaffung und Lagerung der Versuchskomponenten*: In den Versuchsprotokollen fehlt ebenfalls die Information, wo man die entsprechenden Versuchskomponenten im realen Labor herbekommt. So werden keine Angaben darüber gemacht, in welchem Behälter sich die benötigten Substanzen befinden. Des Weiteren fehlen Informationen über die Lagerungsbestimmungen der Versuchskomponenten und die Inhalte der einzuhaltenden Sicherheitsbestimmungen.

- *Zwischenschritte*: Eventuelle Zwischenschritte zu den einzelnen Tätigkeiten im Versuchsprotokoll müssen angegeben werden. So fehlt hier zum Beispiel die Angabe der Zwischenschritte, dass die Agarose mit der Waage und das der TBE-Puffer mit dem Messbecher abgemessen wird.

- *Zeitangaben*: Zu jedem Versuchsschritt müssen genaue Zeitangaben gemacht werden, das heißt es muss zum Beispiel angegeben werden, wie lange das Gemisch aus TBE-Puffer und Agarose aufgekocht werden muss.

- *Visuelle Informationen*: Bezüglich der Substanzen sind generell mindestens die Farbe und der Aggregatzustand anzugeben. Reagieren Substanzen miteinander, so ändert sich unter Umständen deren äußeres Erscheinungsbild. Das kann sich zum Beispiel in der Farbe oder dem chemischen Zustand der Substanz zeigen. Ein Beispiel dafür ist die Oxidation von Kupfer zu Kupferoxid und die damit verbundene Farbänderung von Hellrot nach Dunkelrot. Oder die Substanz, wie zum Beispiel aufkochendes Wasser, geht vom flüssigen in den gasförmigen Zustand über.

- *Formeln und Regeln zu den Reaktionen und Abläufen*: In den Versuchsprotokollen wird das theoretische Hintergrundwissen zu den Reaktionen und Abläufen nicht beschrieben. Dieses ist jedoch wesentlich für das Verständnis des Versuchs. Ohne Kenntnis der Formeln und Regeln der Reaktionen und Abläufe kann das Simulationsmodell nicht erstellt werden.

- *Hierarchie der Arbeitsschritte (parallele vs. sequentielle Abläufe)*: Versuchsprotokolle berücksichtigen nicht, dass einige Arbeitsschritte auch vertauscht werden können (Variationen in den Versuchsabläufen). Des Weiteren wird in den Versuchsprotokollen keine Hierarchie der Arbeitsschritte angegeben, das heißt es fehlt die Angabe, wie die einzelnen Arbeitsschritte voneinander abhängen.

- *Modularisierung der Versuchsprotokolle*: Die Versuchsprotokolle setzen sich im Allgemeinen aus mehreren (generischen) Teilversuchen zusammen. Diese Teilversuche müssen identifiziert und können als eigenständige Versuche betrachtet werden

(siehe Kapitel 6.1). Außerdem werden zur Durchführung der Versuche zahlreiche Fertigkeiten benötigt, die ebenfalls identifiziert werden müssen. Die Modularisierung von Versuchsprotokollen ist besonders dann wichtig, wenn neue Versuche dem virtuellen Labor hinzugefügt werden, um dann auf bereits implementierte Teilversuche und Versuchskomponenten zurückgreifen zu können.

Ergebnis dieser Aufbereitung sind die um Zusatzinformationen erweiterten Versuchsprotokolle, das heißt Protokolle, die exakter sind als üblich, und daher als *annotierte Versuchsprotokolle*^Artefakt bezeichnet werden (siehe dazu auch die Checkliste zur Bearbeitung der Versuchsprotokolle in Anhang B.3). Anschließend werden die *annotierten Versuchsprotokolle*^Artefakt von den *Fachdidaktikern*^Rolle geprüft, das heißt es wird nach bestehenden Lücken und Fehlern in den Versuchsabläufen gesucht. Des Weiteren werden Arbeitsschritte vereinfacht, falls dies möglich ist.

Dem *Fachexperten*^Rolle und *Fachdidaktiker*^Rolle sollten außerdem die Grenzen der *Virtualisierung* der Versuchsprotokolle bekannt sein, das heißt die Probleme, die bei Umsetzung des Versuchsprotokolls und der zugehörenden Arbeitsflächen, Laborgeräte, Behälter und Substanzen auf ein Computer-System entstehen. Die Probleme der Virtualisierung zeigen sich in den folgenden Unterschieden zwischen der praktischen (das heißt realen) und der virtuellen Versuchsdurchführung:

- *Spezifikation des Aufbaus der Lerneinheit*: In der Aktivität *Bestimme Anzahl und Ausstattung der Laborräume*^Aktivität (siehe Abschnitt 7.2.3.2) wurden die benötigten Laborräume identifiziert und im virtuellen Labor angeordnet. Für die Spezifikation des Aufbaus der Lerneinheit wird festgelegt, auf welchen Arbeitsflächen sich welche Versuchskomponenten befinden. Dabei ist insbesondere zu berücksichtigen, dass sich die Versuchskomponenten nicht unbedingt so auf die Arbeitsflächen verteilen lassen, wie es im realen Labor möglich ist. Ursache dafür kann sein, dass der Platz auf den virtuellen Arbeitsflächen aufgrund des eingeschränkten Sichtbereichs des Bildschirms nicht ausreicht (die realen Arbeitsflächen entsprechen in der Regel nicht der 4:3-Aufteilung des Bildschirms). Das virtuelle Labor muss dann um weitere Arbeitsflächen ergänzt werden. Es ist auch möglich, dass das virtuelle Laborgerät trotz des richtigen Maßstabes einfach zu groß für die Arbeitsfläche ist. In diesem Fall muss die bestehende Arbeitsfläche erweitert werden (siehe Konzept des Flickenteppichs in *Bestimme Möglichkeiten zum Wechsel der Arbeitsfläche*^Aktivität in Abschnitt 7.2.3.3).

- *Größenverhältnisse der Versuchskomponenten*: Zur Erstellung der realitätsgetreuen Abbilder der Versuchskomponenten ist auf die korrekten Größenverhältnisse der Laborgeräte, Behälter und Substanzen zu achten. Dazu muss möglichst frühzeitig bestimmt werden, wie groß das größte Laborgerät im virtuellen Labor ist. Der Maßstab dieses Laborgerätes kann dann auf die anderen Versuchskomponenten übertragen werden. Insgesamt darf der Maßstab der Versuchskomponenten aber auch nicht zu groß gewählt werden, weil ansonsten kleine Behälter (wie beispielsweise Eppendorfgefäße) nicht mehr sinnvoll auf der Benutzungsoberfläche dargestellt werden können. Dieser Fall tritt dann ein, wenn der Größenunterschied zwischen der kleinsten und der größten

Versuchskomponente zu hoch ist. Für zu kleine Versuchskomponenten wird dann eine Ausnahme gemacht und deren Maßstab reduziert, das heißt auf dem Bildschirm größer dargestellt. Bezüglich der Größenverhältnisse der Versuchskomponenten ist auch zu prüfen, ob Laborgeräte nicht deshalb so groß sind, weil sie längst veraltet sind und durch modernere und kleinere ausgetauscht werden müssten. Es ist dann zu überlegen, inwiefern es überhaupt Sinn macht, das alte Laborgeräte im virtuellen Labor noch nachzubilden. Dies gilt insbesondere dann, wenn das alte Laborgerät besonders groß und damit der Größenunterschied zur kleinsten Versuchskomponente sehr hoch ist.

- *Probenanzahl*: Werden in der realen Versuchsdurchführung eine sehr hohe Anzahl an Proben (zum Beispiel 30 Stück) erstellt, so können die Proben im virtuellen Labor nicht mehr sinnvoll gehandhabt und auf der Arbeitsfläche dargestellt werden. In der Realität ist die hohe Probenanzahl im Allgemeinen jedoch zwingend notwendig, damit bedingt durch die Variationen in der Ausbeute der Proben überhaupt ein signifikantes Versuchsergebnis ermittelt werden kann. Für das virtuelle Labor ist dann zu prüfen, ob von den Variationen in der Ausbeute der Proben abstrahiert und idealisiert werden kann, so dass auch eine geringere Probenanzahl zur erfolgreichen Durchführung des Versuchs ausreicht.

- *Alternativen der Versuchsdurchführung*: Gegebenenfalls gibt es Alternativen in der realen Versuchsdurchführung, die so nicht dem *annotierten Versuchsprotokoll*^{Artefakt} entnommen werden können beziehungsweise enthalten sind. So ist es zum Beispiel möglich, dass ein Laborgerät durch einen Laborraum ersetzt werden kann oder umgekehrt. Ein Beispiel dafür ist der Heizblock und der 37°-Raum. Wird der Heizblock auf $37°C$ eingestellt, so kann ein Versuch, der eine Temperatur von $37°C$ benötigt, sowohl mit Hilfe des Heizblocks als auch im 37°-Raum durchgeführt werden. In diesem Fall muss im Gesamtkontext aller Versuche entschieden werden, ob das Laborgerät, der Laborraum oder beides im virtuellen Labor umgesetzt werden soll. Einen Laborraum im virtuellen Labor hinzuzufügen macht allerdings nur dann Sinn, wenn dieser auch in mehreren Versuchen genutzt wird. Die Verwendung eines Laborraums anstelle eines Laborgerätes ist insbesondere dann sinnvoll, wenn bei Verwendung des Laborgerätes der Größenunterschied zwischen der kleinsten und der größten Versuchskomponente zu hoch ist beziehungsweise das Laborgerät allgemein zu groß für das virtuelle Labor ist. Bezüglich der Alternativen der Versuchsdurchführung ist es aber auch möglich, dass andere Laborgeräte oder Behälter, als die im *annotierten Versuchsprotokoll*^{Artefakt} angegebenen, verwendet werden können. So lassen sich zum Beispiel Laborgeräte durch andere ersetzen, wenn diese veraltet sind. Behälter können durch andere ersetzt werden, wenn diese zum Beispiel für das Mischen von Substanzen gebraucht werden. Es ist dann irrelevant, ob die Substanzen in dem Messbecher oder dem Messzylinder zusammengegeben werden.

- *Vereinheitlichung der Versuchsprotokolle*: Es ist möglich, dass die Angaben zu den Geräteeinstellungen in verschiedenen Versuchsprotokollen nicht einheitlich sind, obwohl die Unterschiede in den Geräteeinstellungen für das Versuchsergebnis nicht signifikant sind. So ist es denkbar, dass die Einstellung der Umdrehungszahl einer Zentrifuge

> in den Versuchsprotokollen mit 13.000, 14.000 oder auch 15.000 Umdrehungen pro Minute angegeben wird, obwohl jeweils auch einer der anderen Werte erlaubt gewesen wäre (insbesondere also auch jedesmal 14.000 Umdrehungen pro Minute). Sind Geräteeinstellungen nicht entscheident für das Versuchsergebnis, so sind diese in den Versuchsprotokollen zu vereinheitlichen, das heißt die Versuchsprotokolle sind aufeinander abzustimmen und zu optimieren. Die Vereinheitlichung ist insbesondere in Hinsicht auf die Erstellung der Laborgeräte wichtig, um zu entscheiden, welche Einstellungsmöglichkeiten an dem jeweiligen Laborgerät modelliert werden müssen. So kann zum Beispiel eine Zentrifuge im realen Labor einen Drehregler zur kontinuierlichen Einstellung der Umdrehungszahl haben, während in den Versuchen des virtuellen Labors nur die beiden Einstellungen von entweder 7.000 oder 14.000 Umdrehungen pro Minute benötigt werden. Für die virtuelle Zentrifuge kann der Drehregler dann so modelliert werden, dass er nur diese beiden diskreten Werte annehmen kann.

Bezüglich der Virtualisierung der Versuchsprotokolle ist es wichtig, dass die genannten Probleme von dem *Fachexperten*[Rolle] und dem *Fachdidaktiker*[Rolle] verstanden und nachvollzogen werden, um den *Informatikern*[Rolle] gegebenenfalls gezielt Hilfestellungen geben zu können. So können bereits bei der Erstellung der *annotierten Versuchsprotokolle*[Artefakt] diese um Hinweise zur Virtualisierung ergänzt werden.

Wichtig im Umgang mit den Versuchsprotokollen beziehungsweise den *annotierten Versuchsprotokollen*[Artefakt] ist auch das vorliegende Format. In der Regel werden die Versuchsprotokolle von den *Fachexperten*[Rolle] mit Hilfe eines gängigen Textverarbeitungssystems, wie zum Beispiel *Mircosoft Word*[8], erstellt und in dessen Dateiformat gespeichert. Dieses Dateiformat kann jedoch nicht direkt weiterverarbeitet werden. Daher müssen die Versuchsprotokolle zunächst konvertiert werden. Anschließend können die Versuchsprotokolle in ein Autorensystem beziehungsweise die verwendete Entwicklungsumgebung durch einfaches Einfügen oder mit Hilfe spezieller Werkzeuge integriert und beispielsweise zur Erstellung eines Anleitungsfensters verwendet werden. Des Weiteren ist die Erstellung einer zusätzlichen längeren Version der Versuchprotokolle für den Ausdruck auf Papier und für das Online-Handbuch denkbar (Hasler u. a., 2001).

Zur besseren Veranschaulichung und zur Visualisierung der Arbeitsabläufe und des Versuchsaufbaus wird außerdem auf der Grundlage der *annotierten Versuchsprotokolle*[Artefakt] ein *Drehbuch*[Artefakt] erstellt (siehe auch Jarz, 1997, Seite 261 bis 296). In dem *Drehbuch*[Artefakt] werden die Lerneinheiten in einer nicht-formalen Spezifikationssprache festgehalten und besteht im Wesentlichen aus einer graphischen Skizzierung von Aufbau und Ablauf der Lerneinheit. So wurde zum Beispiel in *GenLab* die Spezifikation von Aufbau und Ablauf der Versuche mit Hilfe eines *Drehbuchs*[Artefakt] durchgeführt. Aus den Erfahrungen von *GenLab* ist für die Erstellung des *Drehbuchs*[Artefakt] wie folgt vorzugehen: Zunächst wird der Ablauf des Versuchs skizziert (*dynamische Sicht*). Dazu werden die einzelnen Arbeitsschritte des *annotierten Versuchsprotokolls*[Artefakt] in skizzierter graphischer Form dargestellt. Dabei ist es

[8] Hersteller: *Microsoft* (siehe: http://www.microsoft.com/)

besonders wichtig, dass die Notation zur Skizzierung des Arbeitsablaufs intuitiv verständlich ist und keiner aufwendigen Erklärung bedarf. So ist aus Abbildung 7.15 zu ersehen, dass zu dem aufgekochten Gel im Messbecher mit Hilfe der Mikropipette $10\mu l$ Ethidiumbromid hinzuzugeben sind. Anschließend wird der Messbecher geschüttelt. Dann wird zunächst der Gelkamm in den Gelträger gelegt und anschließend das aufgekochte Gel aus dem Messbecher in den Gelträger gegeben.

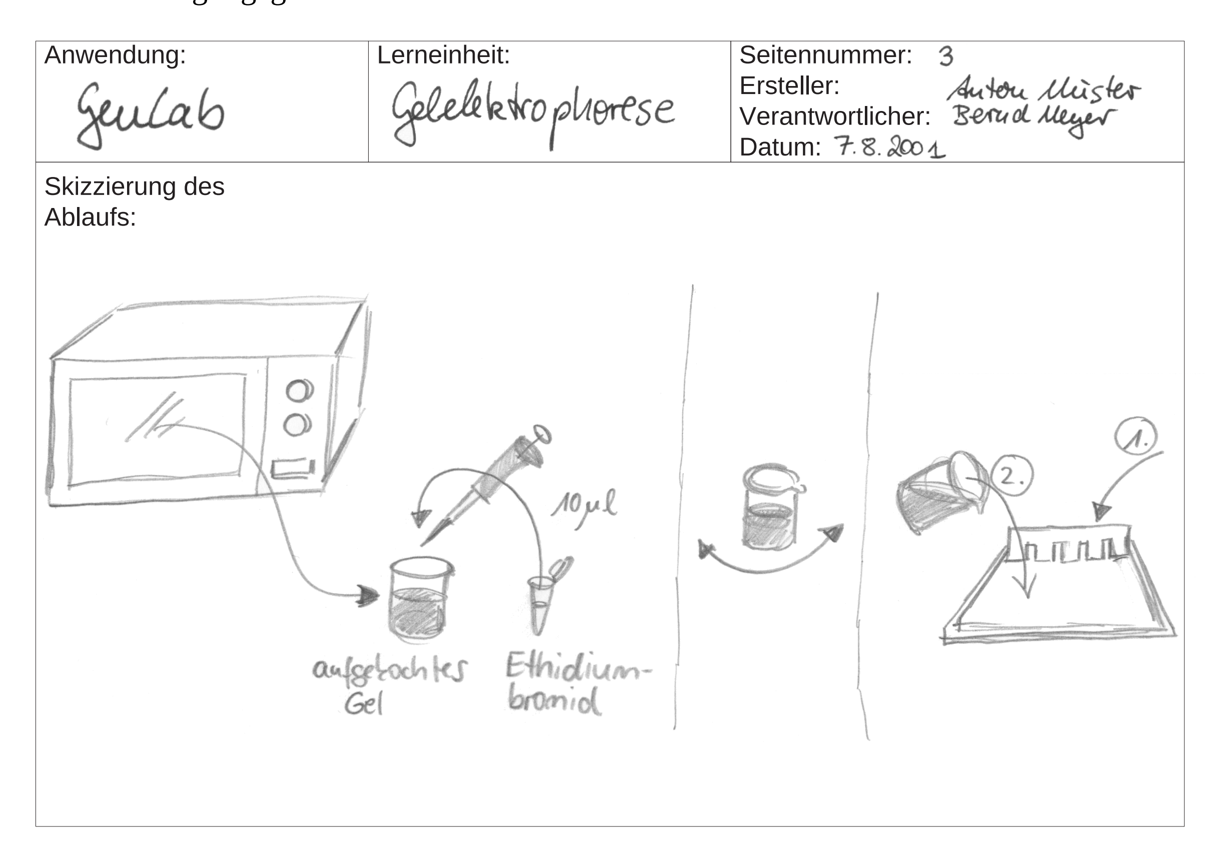


Abbildung 7.15.: Beispiel einer Drehbuchseite zur Darstellung des Versuchsablaufs

Nach der Skizzierung des Ablaufs wird der Aufbau des Versuchs festgelegt (*statische Sicht*). Dazu wird die Anzahl und Ausstattung der benötigten Arbeitsflächen anhand der dynamischen Sicht bestimmt, das heißt es werden die einzelnen Drehbuchseiten zum Ablauf des Versuchs durchgegangen und die Arbeitsflächen gezählt. Anschließend werden die Laborgeräte, Behälter und Substanzen auf den Arbeitsflächen angeordnet und geprüft, ob der Platz und damit die Anzahl der Arbeitsflächen ausreicht. Steht auf den Arbeitsflächen nicht hinreichend Platz zur Verfügung, so muss das virtuelle Labor entsprechend erweitert werden (siehe dazu oben die Spezifikation des Aufbaus der Lerneinheit und die Aktivität *Bestimme Anzahl und Ausstattung der Laborräume*^{Aktivität} in Abschnitt 7.2.3.2). Des Weiteren ist zur Spezifikation des Aufbaus der Lerneinheit festzulegen, welche Versuchskomponenten sich in der Transportleiste befinden sollen. So ist aus Abbildung 7.16 ersichtlich, dass die Arbeitsfläche mit der Mikrowelle, dem Gelträger und dem Gelkamm bereits nahezu vollständig belegt ist. Des Weiteren befindet sich das Eppendorfgefäß mit dem Ethidiumbromid in der Transportleiste (vergleiche dazu Abbildung 2.2).

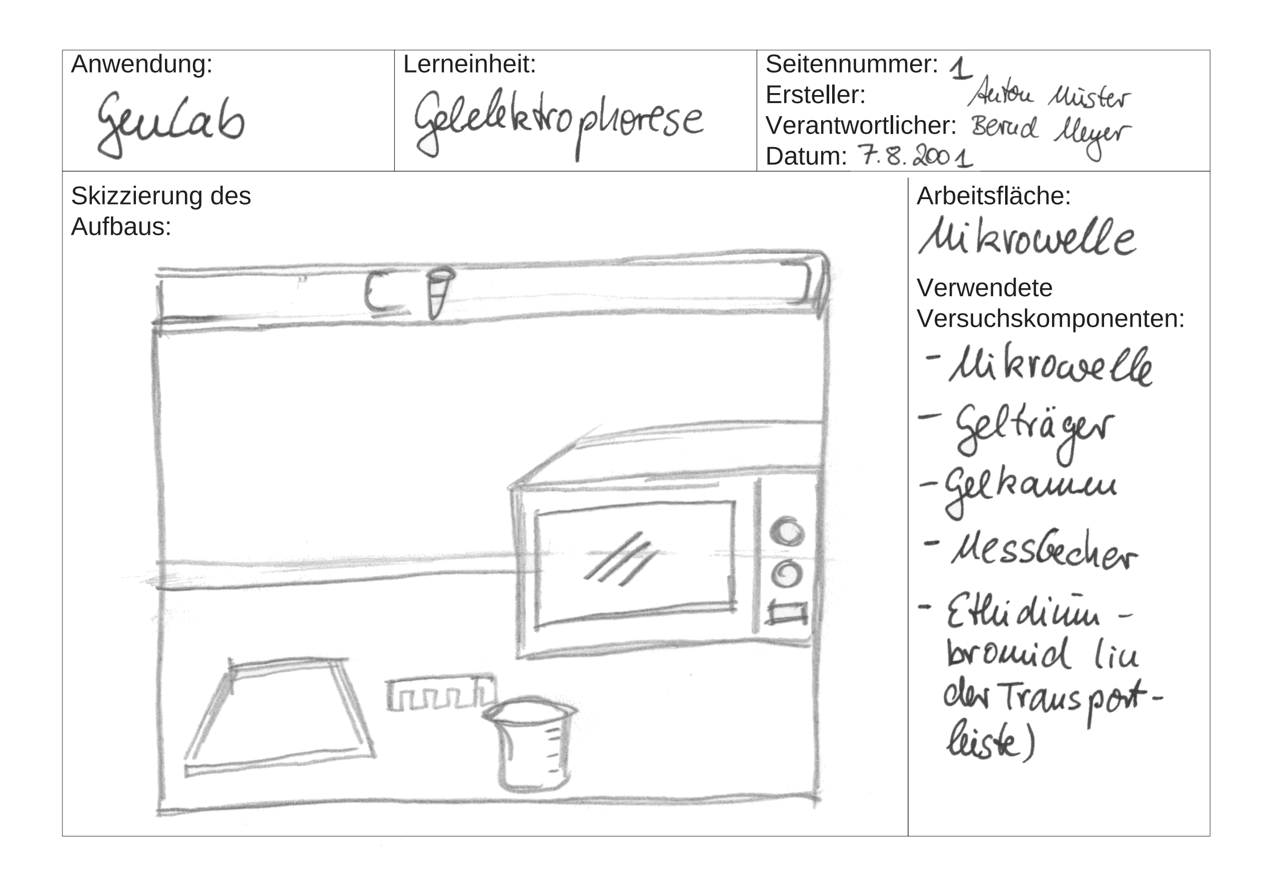


Abbildung 7.16.: Beispiel einer Drehbuchseite zur Darstellung des Versuchsaufbaus

Die Darstellungen des Ablaufs und Aufbaus einer Lerneinheit sind in ein allgemeines Muster einer Drehbuchseite (nach Yass, 2000, Seite 138 und 139) eingebettet (siehe Abbildungen 7.15 und 7.16). Im oberen Teil einer Drehbuchseite befindet sich das Feld *Anwendung*, in dem der Name des zu erstellenden virtuellen Labors eingetragen wird. Mit der *Lerneinheit* wird der Name des Versuches oder der Fertigkeit bezeichnet. Für organisatorische Zwecke wird außerdem die *Seitennummer*, der *Ersteller*, der *Verantwortliche* und das *Datum* zu der Drehbuchseite festgehalten. Im unteren Teil einer Drehbuchseite für den Ablauf einer Lerneinheit (siehe Abbildung 7.15) werden die einzelnen Arbeitsschritte skizziert. In der Regel werden zur Darstellung des Ablaufs einer Lerneinheit mehrere Drehbuchseiten benötigt, die dann mit einer fortlaufenden Seitennummer versehen werden. Im linken unteren Teil einer Drehbuchseite für den Aufbau einer Lerneinheit (siehe Abbildung 7.16) wird die Position der Laborgeräte, Behälter und Substanzen auf der Arbeitsfläche des virtuellen Labors festgelegt. Des Weiteren kann die Skizzierung des Aufbaus der Lerneinheit weitere Zusatzinformationen zu den *verwendeten Versuchskomponenten* und der *Arbeitsfläche* enthalten. Diese werden im rechten unteren Feld der Drehbuchseite eingetragen. Für jede Arbeitsfläche der Lerneinheit wird eine entsprechende Drehbuchseite zur Skizzierung des Aufbaus der Versuchskomponenten erstellt. Außerdem sind aus dem *Drehbuch*^{Artefakt} neben Aufbau und Ablauf der Lerneinheiten auch die Farbgebung der Arbeitsfläche, sowie der Laborgeräte, Behälter und Substanzen zu entnehmen.

Wünschenswert ist außerdem ein entsprechendes Entwicklungswerkzeug, dass die Erstellung des *Drehbuchs*[Artefakt] geeignet unterstützt (Nagl u. a., 1999, Seite 103). Dabei werden zum Beispiel die Felder der Drehbuchseite wie Anwendung, Lerneinheit, Seitennummer, Ersteller, Verantwortlicher und Datum vom Werkzeug ausgefüllt. Die Skizzierung von Aufbau und Ablauf der Lerneinheit kann zum Beispiel mit Hilfe eines Graphiktabletts erfolgen. Dafür können gegebenenfalls auch Miniaturansichten zu den Versuchskomponenten verwendet werden. Gegenüber einem handskizziertem *Drehbuch*[Artefakt] bietet das Werkzeug die Möglichkeit, auf einfache Weise Änderungen und Anpassungen am Versuchsaufbau und -ablauf vorzunehmen. Des Weiteren können die einzelnen Drehbuchseiten zu den Lerneinheiten mit Hilfe dieses Werkzeuges verwaltet werden.

Eine alternative Möglichkeit zur Darstellung der Versuchsabläufe sind Aktivitätsdiagramme. Dies setzt allerdings voraus, dass Aktivitätsdiagramme und die entsprechende Notation von allen *Projektmitarbeitern*[Rolle] und insbesondere den *Fachexperten*[Rolle] und *Fachdidaktikern*[Rolle] verstanden und akzeptiert werden (siehe dazu *Erhebe die Anwendungsfälle*[Aktivität] in Abschnitt 7.2.1.1). Wesentlicher Vorteil der Aktivitätsdiagramme gegenüber dem *Drehbuch*[Artefakt] ist die standardisierte Notation. Allerdings bleiben die gestalterischen Aspekte des virtuellen Labors bei der Erstellung von Aktivitätsdiagrammen unberücksichtigt.

7.3.4.2. Berücksichtige mögliche Variationen der Versuchsabläufe

Entscheidend für die Akzeptanz des virtuellen Labors ist, dass die Spezifikation der Versuchsabläufe auch von externen *Fachexperten*[Rolle] geprüft wird. Nur so lassen sich Varianten in der Durchführung eines Versuches an anderen Universitäten oder Instituten ermitteln und geeignete Abstraktionen finden. Orientieren sich die Versuchsabläufe des virtuellen Labors nach der konkreten Durchführung der Versuche an einer einzigen Universität oder einem einzigen Institut, so besteht die Gefahr, dass das virtuelle Labor an anderen Universitäten und Instituten mit abweichendem Versuchsprotokoll nur schwer oder gar nicht akzeptiert wird. In diesem Zusammenhang ist auch zu prüfen, ob die Laborgeräte, die im virtuellen Labor verwendet werden, nicht bereits in der Realität veraltet sind und durch modernere ersetzt wurden. Auch besteht die Möglichkeit, dass ein vollständiger Versuch durch ein speziell dafür entwickeltes Laborgerät ersetzt wurde beziehungsweise werden kann.

7.3.5. Spezifiziere das Hintergrundwissen zur Lerneinheit

Aus dem Metaobjektmodell für virtuelle Labore in Kapitel 6 ist ersichtlich, dass zu jedem (Teil-)Versuch und zu jeder Versuchskomponente ein entsprechendes theoretisches Hintergrundwissen (siehe Theorie in Kapitel 6.2) gehört (siehe dazu Hasler und Schlattmann, 2001, Seite 13 bis 20). Dieses wird von den *Fachexperten*[Rolle] identifiziert und im Detail bestimmt. Außerdem wird das Hintergrundwissen von den *Fachdidaktikern*[Rolle] aufbereitet.

Zur Beschreibung und Darstellung des Hintergrundwissens können verschiedene Medien eingesetzt werden. So können Texte zur Beschreibung des theoretischen Hintergrundwissens zu den Versuchen und Versuchskomponenten verwendet werden. Graphiken, wie zum

Beispiel Zeichnungen und digitalisierte Photos, werden zur Veranschaulichung von Reaktionsgleichungen und zur Darstellung der Versuchskomponenten verwendet. Die Ansicht von allen Seiten mit Hilfe von 3D-Modellen erlaubt es dem Lerner, die Laborgeräte, Behälter und die molekulare Struktur der Substanzen von allen Seiten zu betrachten und interaktiv erfahrbar zu machen. Außerdem kann das Hintergrundwissen durch Videosequenzen und Animationen ergänzt und mit Audiosequenzen vertont werden (siehe dazu *Ermittle benötigte Medien*[Aktivität] in Kapitel 7.5.2).

Das theoretische Hintergrundwissen zu einem Versuch besteht in *GenLab* aus einer textuellen Übersicht und Beschreibung des Versuchsablaufs mit ergänzenden Animationen und dem Versuchsprotokoll. Zu jedem Laborgerät gibt es ein digitalisiertes Photo des realen Laborgerätes, ein 3D-Modell vom Laborgerät zur Ansicht von allen Seiten, eine kurze Beschreibung des Laborgerätes sowie eine ausführliche Bedienungsanleitung inklusive der Sicherheitsbestimmungen. Das Hintergrundwissen zu einem Behälter beziehungsweise Zubehör besteht aus einem digitalisierten Photo des realen Behälters beziehungsweise Zubehörs, einem 3D-Modell und eine kurze textuelle Beschreibung. Zu jeder Substanz ist in *GenLab* die Strukturformel als Graphik und eine textuelle Beschreibung der Substanz enthalten. Des Weiteren werden zu jeder Substanz gemäß dem Materialblatt weitere Merkmale genannt, wie zum Beispiel andere Bezeichnungen der Substanz, die Summenformel, Farbe, Geruch, Form, molare Masse, Dichte (bei $20°C$), Löslichkeit in Wasser (bei $20°C$), Schmelzpunkt, Siedepunkt, Sicherheitshinweise, sowie bei Substanzgemischen die Konzentration, das heißt die Zusammensetzung und sonstige Informationen zum Gemisch.

Existiert ein Werkzeug zur Integration des Hintergrundwissens in das virtuelle Labor, so kann die Integration des Hintergrundwissens von einem *Fachexperten*[Rolle] oder einem *Fachdidaktiker*[Rolle] vorgenommen werden (vergleiche das Werkzeug *Laboratory Information Network Authorware* in Aden u. a., 1999, Seite 42). Bei einem separaten Teilsystem für das Hintergrundwissen, wie in *GenLab* als Online-Handbuch, kann die Integration des Hintergrundwissens von einem *Medienspezialisten*[Rolle] durchgeführt werden (siehe dazu *Berücksichtige die Anbindung weiterer Komponenten und Teilsysteme*[Aktivität] in Abschnitt 7.3.3.7). Es ist aber auch denkbar, das Hintergrundwissen direkt in die Benutzungsoberfläche der explorativen Lehr- und Lernumgebung des virtuellen Labors mit einzubinden und dort in einem separaten Fenster darzustellen. Stellt die Integration des Hintergrundwissens ein aufwendigeres softwaretechnisches Problem dar, so wird dies von den *Informatikern*[Rolle] übernommen. Dies ist zum Beispiel dann der Fall, wenn zur Speicherung des Hintergrundwissens eine Datenbank verwendet wird und kein entsprechendes Werkzeug zum Einspielen des Hintergrundwissens in die Datenbank existiert.

7.4. Analyse und Design

Die Abbildung 7.17 zeigt den Workflow *Analyse und Design*[Workflow] des Vorgehensmodells für virtuelle Labore (vergleiche Kruchten, 2000, Seite 178). Die Hauptaufgabe des Workflows ist die Festlegung der *Architektur*[Artefakt] des virtuellen Labors. Dazu sind die software-technischen Aspekte des Simulators und des Tutorsystems zu berücksichtigen.

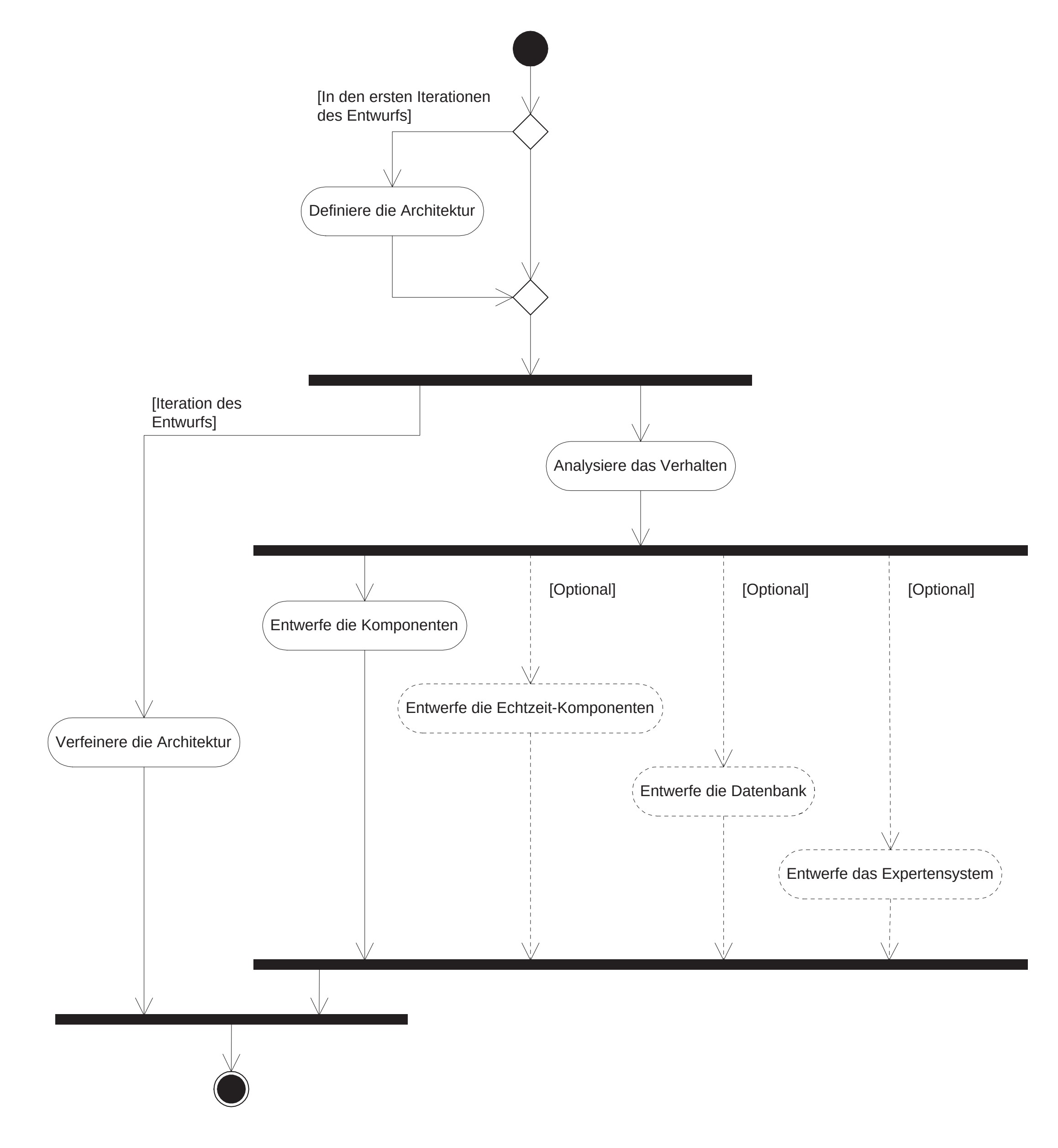


Abbildung 7.17.: Der Workflow zu Analyse und Design

In den ersten Iterationen der Entwurfsphase ist die *Architektur*[Artefakt] des virtuellen Labors festzulegen. Anschließend wird die *Architektur*[Artefakt] verfeinert und das Verhalten des virtuellen Labors, wie es durch die Anwendungsfälle im *Anwendungsfall-Modell*[Artefakt] festgehalten ist, umgesetzt. Dazu werden die einzelnen Komponenten des virtuellen Labors entworfen. Sind im virtuellen Labor Echtzeit-Komponenten vorgesehen, so werden diese parallel zu den anderen Komponenten definiert. Außerdem besteht die Möglichkeit das virtuelle Labor an eine Datenbank oder ein Expertensystem anzubinden.

Die Aktivitäten des Workflows *Analyse und Design*[Workflow] werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Definiere die Architektur*[Aktivität] (siehe Kapitel 7.4.1),
- *Verfeinere die Architektur*[Aktivität] (siehe Kapitel 7.4.2),
- *Analysiere das Verhalten*[Aktivität] (siehe Kapitel 7.4.3) und
- *Entwerfe die Komponenten*[Aktivität] (siehe Kapitel 7.4.4),
  sowie die optionalen Aktivitäten
- *Entwerfe die Echtzeit-Komponenten*[Aktivität] (siehe Kapitel 7.4.5),
- *Entwerfe die Datenbank*[Aktivität] (siehe Kapitel 7.4.6) und
- *Entwerfe das Expertensystem*[Aktivität] (siehe Kapitel 7.4.7).

7.4.1. Definiere die Architektur

Die *Architektur*[Artefakt] des virtuellen Labors ist ein objektorientiertes Framework (siehe Elfreich, 1999), welches im Wesentlichen vorgegeben wird. Daher besteht die Definition der *Architektur*[Artefakt] (siehe Abbildung 7.18) hauptsächlich in der Auswahl von bereits identifizierten Konzepten (vergleiche Hasler und Schlattmann, 2001, Seite 24), die für die Erstellung virtueller Labore relevant sind. Diese Aktivität wird von den *Designern/Entwicklern*[Rolle] unter Rücksprache mit dem *Projektleiter*[Rolle] durchgeführt.

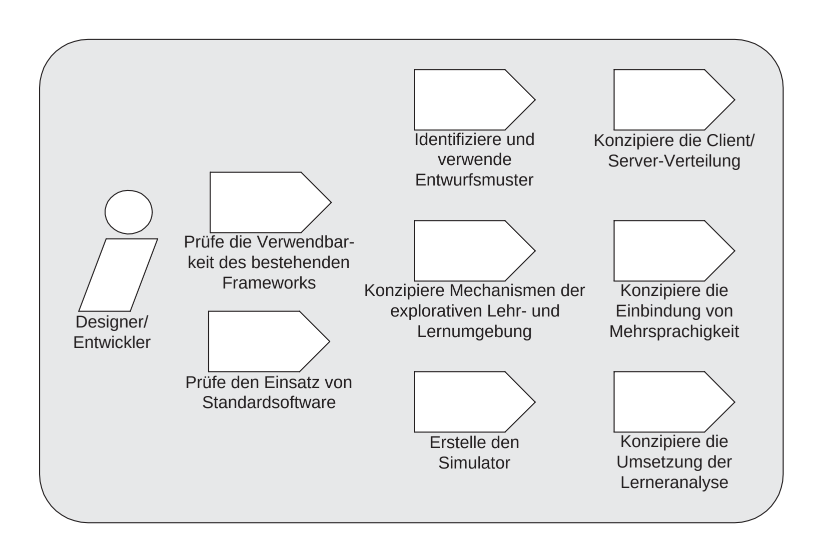


Abbildung 7.18.: Detailansicht der Aktivität Definiere die Architektur

Außerdem sind sämtliche Entscheidungen der Aktivität *Definiere das virtuelle Labor*[Aktivität] (siehe Kapitel 7.2.3) und *Erstelle das didaktische Konzept*[Aktivität] (siehe Kapitel 7.3.3) wieder aufzugreifen. Dazu gehört insbesondere die Einbindung einer Arbeitsflächenverwaltung und einer Transportleiste sowie das dynamische Laden und Speichern von Versuchen.

7.4.1.1. Prüfe die Verwendbarkeit des bestehenden Frameworks

Existiert bereits ein Framework für virtuelle Labore, so ist zu prüfen, inwiefern das bestehende Framework für das neue virtuelle Labor angepasst werden kann oder zu erweitern ist. Das existierende Framework dient dabei als Grundlage für die weitere Entwicklung. Die Abbildung 7.19 gibt einen Überblick über die allgemeine Struktur des Frameworks für virtuelle Labore (vergleiche Hasler und Schlattmann, 2001, Seite 24).

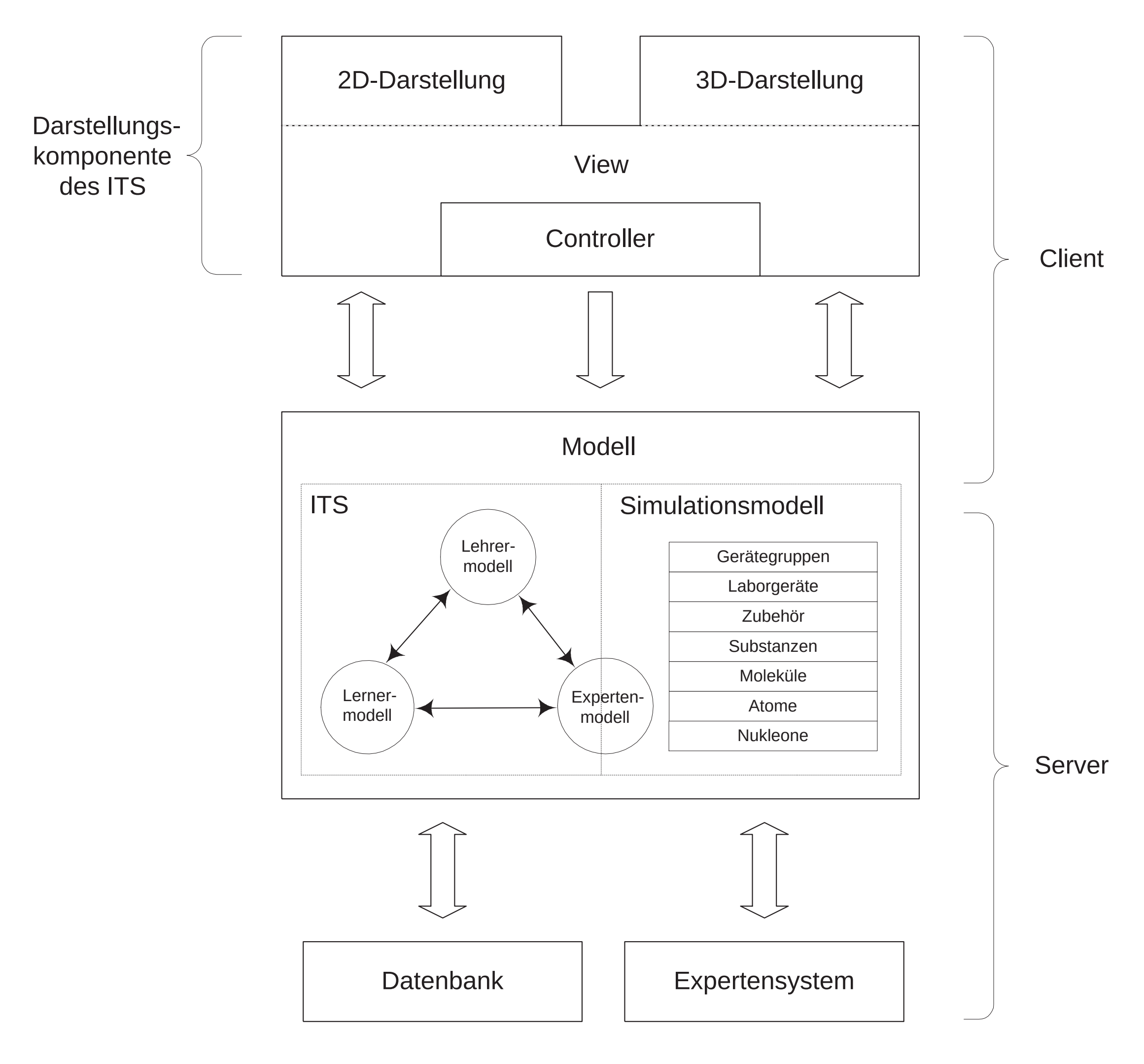


Abbildung 7.19.: Schichtenmodell des Frameworks

7.4.1.2. Identifiziere und verwende Entwurfsmuster

Grundlegendes Prinzip der *Architektur*[Artefakt] beziehungsweise des Frameworks ist das *Model–View–Controller–Konzept* (*MVC–Konzept*). Durch das *Model–View–Controller*–Konzept werden die *Modelle* (*model*), ihre *Bildschirmrepräsentationen* (*view*) und die *Benutzerinteraktionen* (*controller*) voneinander gekapselt und so eine hohe Wiederverwendbarkeit

und Änderbarkeit der Komponenten ermöglicht (Gamma u. a., 2000, Seite 4 f.). Ist eine Aktualisierung des Designs der Benutzungsoberfläche notwendig, weil sich beispielsweise die Benutzungsoberfläche des zugrunde liegenden Betriebssystem geändert hat, so kann dieses durch einfaches Austauschen der Bildschirmrepräsentationen zu den Modellen realisiert werden. Die Lerneinheiten, das zugrunde liegende Simulationsmodell sowie das *didaktische Konzept*^Artefakt des virtuellen Labors können also wiederverwendet werden (vergleiche Ateyeh u. a., 1999, Seite 5). Werden die graphischen Änderungen an der Benutzungsoberfläche nicht nachgezogen, so wirkt das virtuelle Labor bereits nach wenigen Jahren veraltet. Des Weiteren ermöglicht das *Model–View–Controller*–Konzept unterschiedliche Sichten auf die Daten des Modells. So können diese zum Beispiel als Liste von Zahlen oder als Diagramm dargestellt werden. Außerdem erlaubt das *Model–View–Controller*–Konzept die Anpassung der Benutzungsoberfläche an unterschiedliche Benutzer, wie zum Beispiel Anfänger und Experte (siehe Preim, 1999, Seite 283). So können verschiedene Schwierigkeitsstufen und Blickwinkel auf die Domäne des virtuellen Labors realisiert werden (siehe *Bestimme Schwierigkeitsstufen und Blickwinkel auf die Domäne*^Aktivität in Abschnitt 7.2.3.6). Das *Model–View–Controller*–Konzept wird für sämtliche Elemente der Benutzungsoberfläche des virtuellen Labors angewendet. Dazu zählen insbesondere die Laborgeräte, Behälter und Substanzen, aber auch die Arbeitsflächen und Steuerungselemente des virtuellen Labors.

Für den einfacheren Austausch der kompletten Benutzungsoberfläche kann zudem noch das Entwurfsmuster *Abstract Factory* (siehe Gamma u. a., 2000, Seite 87 ff.) angewendet werden. Mit diesem Entwurfsmuster ist es möglich, aus einer abstrakten Klasse der Benutzungsoberfläche verschiedene konkrete Klassen zu erzeugen. Dies ermöglicht ein besonders einfaches Austauschen der Bildschirmrepräsentationen zu den Modellen, da diese nur Kenntnis von der abstrakten Klasse haben. So kann mit dem Entwurfsmuster der *Abstract Factory* zum Beispiel die Sprachauswahl zur Laufzeit realisiert werden (siehe *Konzipiere die Einbindung von Mehrsprachigkeit*^Aktivität in Abschnitt 7.4.1.7).

Teil des *Model–View–Controller*–Konzeptes ist das Entwurfsmuster *Observer* (siehe Gamma u. a., 2000, Seite 193). Mit diesem Entwurfsmuster kann eine Komponente des virtuellen Labors Daten und Zustände verwenden, die von einer anderen Komponente bereitgestellt werden. Wichtig ist dabei die Gewährleistung der Datenkonsistenz zwischen den Komponenten des virtuellen Labors, die durch entsprechenden Austausch und Aktualisierung der Daten und Zustände garantiert wird. Die Komponenten des virtuellen Labors sind durch das Entwurfsmuster *Observer* lose gekoppelt, das heißt eine Komponente des virtuellen Labors hat keine Kenntnis über die Details der verbleibenden Komponenten. Insbesondere ist nicht im voraus bekannt, welche Komponenten die Daten aktualisieren müssen (vergleiche Elfreich, 1999, Seite 33 f.). Diese Verknüpfungen sind einzeln zu erstellen und von der Domäne des virtuellen Labors abhängig.

Im Zusammenhang mit dem *Model–View–Controller*–Konzept lassen sich noch weitere Entwurfsmuster bei der Entwicklung virtueller Labore einsetzen (siehe Elfreich, 1999, Seite 33). Bestehen graphische Objekte aus mehreren Teilen, so lassen sich diese mit dem Entwurfsmuster *Composite* realisieren. Mit dem Entwurfsmuster *Strategy* können außerdem verschiedene Möglichkeiten bei der Benutzereingabe und der Darstellung durch die Bildschirmrepräsentationen umgesetzt werden. Des Weiteren kann das Entwurfsmuster

Facade (Fassade) angewendet werden, wenn eine Gruppe von Klassen nach außen hin über eine Schnittstelle gekapselt werden soll (siehe Gamma u. a., 2000, Seite 185 ff.).

7.4.1.3. Prüfe den Einsatz von Standardsoftware

Des Weiteren ist bei der Definition der *Architektur*^{Artefakt} zu prüfen, ob Standardsoftware bei der Entwicklung des virtuellen Labors verwendet werden kann. Ist dies der Fall, so ist die *Architektur*^{Artefakt} entsprechend anzupassen und die Standardsoftware zu integrieren. Beispiele für mögliche Standardkomponenten sind:

- *LabView*[9] von *National Instruments* (siehe: www.ni.com/labview/),
- *Simplorer*[10] von *SIMEC*
  (siehe: http://www.simec.com/german/simplorer_index.htm),
- *SMART SPICE* von *SILVACO International* (siehe: http://www.silvaco.com/) und
- *Mathematica* von *Wolfram Research, Inc.*
  (siehe: http://www.wolfram.com/products/mathematica/).

7.4.1.4. Konzipiere Mechanismen der explorativen Lehr- und Lernumgebung

Für die Modellierung der im virtuellen Labor verwendeten Laborgeräte und deren Zusammenspiel sind für das Framework mehrere Aspekte zu berücksichtigen, die als *Mechanismen der explorativen Lehr- und Lernumgebung* bezeichnet werden (siehe Hasler und Schlattmann, 2001, Seite 10). Die wichtigsten Mechanismen sind:

- *Drag & Drop-Mechanismus*: Mit Hilfe des *Drag & Drop*-Mechanismus können die Versuchskomponenten, also die Laborgeräte, Behälter und Substanzen im virtuellen Labor durch Ziehen mit der Computer-Maus von einer Stelle zu einer anderen bewegt werden.
- *Containment*: Einige Laborgeräte, wie zum Beispiel die Zentrifuge oder die Waage, fungieren als Behälter und können andere Laborgegenstände aufnehmen (siehe funktionaler Behälter in Kapitel 6.1).

[9] Findet zum Beispiel Verwendung im *Virtuellen Labor für Regenerative Energien* der Universität Kassel (siehe: http://www.uni-kassel.de/pavi/VirLab/). *LabView* wird dort für den Simulator und die Benutzungsoberfläche eingesetzt.

[10] Wird zum Beispiel im *Virtuellen Praktikum Leistungselektronik* der Technischen Universität Chemnitz (siehe: http://www.infotech.tu-chemnitz.de/∼leielek/le-home.html) verwendet (siehe Kronberg, 1998, Seite 29). Ebenfalls verwendet beziehungsweise erprobt werden *SMART SPICE* und *Mathematica* (siehe Jacob und Kronberg, 1999, Seite 66).

- *Kollisionserkennung*: Bei der Kollisionserkennung wird zum Beispiel kontrolliert, ob der betreffende Laborgegenstand an der gewünschten Stelle abgesetzt werden kann. Kollidiert dieser mit einem anderen Laborgegenstand, so wird beispielsweise die nächste verfügbare freie Stelle auf der Arbeitsfläche gesucht und der Laborgegenstand dort abgesetzt (siehe dazu Heuten, 2001, Seite 18 bis 20).

- *Schwerkraft*: Ein Laborgegenstand kann nicht in der Luft schweben, sondern fällt durch die Schwerkraft wieder auf die Arbeitsfläche zurück.

- *Z-Sortierung*[11]: Bei einer dreidimensionalen Arbeitfläche ist auch die Position des Laborgegenstandes in der Tiefe des Raumes von Bedeutung. So können Laborgeräte, Zubehör und Substanzen nicht nur neben- und übereinander, sondern bei Verwendung der Z-Sortierung auch vor- beziehungsweise hintereinander stehen.

7.4.1.5. Erstelle den Simulator

Grundlage für die Erstellung eines wie in Kapitel 2.6 definierten Simulators, ist das in Kapitel 6 beschriebene Metaobjektmodell für virtuelle Labore. Die anhand des Metaobjektmodells für virtuelle Labore in der Aktivität *Analysiere die Domäne*^Aktivität (siehe Kapitel 7.3.2) identifizierten Bestandteile der Domäne, wie zum Beispiel die verschiedenen Laborgeräte, Behälter und Substanzen, sowie deren Interaktionsbeziehungen zueinander, werden zum Simulationsmodell zusammengefasst (*Konstruktion des Simulationsmodells*) und in einem Simulator implementiert (vergleiche dazu Dick, 2000, Seite 30 f.). Handelt es sich um einen diskreten Simulator, so kann zum Beispiel ein Zustandsdiagramm der Reaktionsabläufe erstellt werden. Liegt eine entsprechende Wissensbasis für die Domäne vor, so kann für die Erstellung des Simulators das transformationelle Vorgehensmodell verwendet werden. Soll im Simulator die Möglichkeit für *Zeitraffer* realisiert werden, so ist in das Framework eine entsprechende *Zeitverwaltung* zu integrieren. Mit einem Zeitraffer besteht die Möglichkeit, die Reaktionsabläufe im virtuellen Labor schneller als in der Realität ablaufen zu lassen. So wurden sämtliche Reaktionsabläufe der gentechnischen Versuche in *GenLab* um den Faktor 60 beschleunigt. Die Navigation im virtuellen Labor und die Bedienung der Laborgeräte entsprechen jedoch weiterhin der Geschwindigkeit, wie in der Realität. Die Aktivitäten des Lerners im virtuellen Labor dauern also aus der Sicht des Frameworks um den Faktor 60 länger. Daher ist im Zusammenhang mit einer *Zeitverwaltung* darauf zu achten, dass die Punktebewertung des *Tutorkonzeptes*^Workflow entsprechend angepasst wird (siehe *Konzipiere die Lerneranalyse*^Aktivität in Abschnitt 7.3.3.4).

7.4.1.6. Konzipiere die Client/Server-Verteilung

Wird das virtuelle Labor als Client/Server-Anwendung konzipiert, zum Beispiel für die Nutzung im Internet, so ist eine sinnvolle Aufteilung der *Architektur*^Artefakt in Client und Server vorzunehmen. In Abbildung 7.19 ist beispielhaft eine mögliche Aufteilung der

[11] Auch: *Z-Order Sortierung* (Heuten, 2001, Seite 4)

Architektur[Artefakt] in Client und Server dargestellt. Demnach besteht der Client aus der Benutzungsoberfläche und gegebenenfalls aus einem Teil des Modells des virtuellen Labors. So ist es zum Beispiel denkbar, dass sich ein Teil des Lernermodells im Client befindet. Dort werden bereits bestimmte Eingabedaten des Lerners analysiert und somit die zu übertragende Datenmenge zum Server reduziert. Auf der Seite des Servers befindet sich dagegen das intelligente Tutorsystem und das Simulationsmodell, sowie gegebenenfalls eine Datenbank und ein Expertensystem. Besonders wichtig bei der Erstellung einer Client/Server-Anwendung ist eine klare und möglichst stabile Schnittstelle zwischen Client und Server. Änderungen am Server sollten keine Änderungen am Client erfordern und umgekehrt. Eine ausführliche Analyse weiterer möglicher Formen von Client/Server-Architekturen befindet sich zum Beispiel in (Knoblich, 1998, Seite 11 bis 28).

7.4.1.7. Konzipiere die Einbindung von Mehrsprachigkeit

Bezüglich der Mehrsprachigkeit muss entschieden werden, wie diese umgesetzt wird. Nach (Heinrich und Schifman, 2000, Seite 35) gibt es folgende Einsatzarten für Mehrsprachigkeit:

- *Getrennte Sprachversionen*: Bei der getrennten Sprachversion werden je nach Anzahl der gewünschten Sprache eine entsprechende Anzahl virtueller Labore erstellt. Damit ist die *Architektur*[Artefakt] des mehrsprachigen virtuellen Labors identisch mit der *Architektur*[Artefakt] ohne Mehrsprachigkeit und eine Umsetzung entsprechend einfach. Letztlich wird dann jedoch nicht nur ein virtuelles Labor entwickelt, sondern mehrere gleichzeitig. Hierbei besteht das Problem, dass Änderungen am virtuellen Labor einer Sprachversion bei allen anderen nachgezogen werden müssen, was sehr aufwendig und fehleranfällig ist.

- *Sprachauswahl am Anfang*: Bei der Sprachauswahl am Anfang kann nach dem Start des virtuellen Labors über ein Menü die gewünschte Sprache ausgewählt werden. Die Einbindung der entsprechenden Medien für die ausgewählte Sprache muss dann also vom virtuellen Labor dynamisch erfolgen. Ein Wechsel zu einer anderen Sprache ist während der Durchführung eines Versuchs dann nicht mehr möglich.

- *Sprachauswahl zur Laufzeit*: Bei dieser Einsatzart kann jederzeit, also auch während der Durchführung eines Versuchs, zwischen den verschiedenen Sprachen gewechselt werden. Dadurch entsteht ein erheblicher Mehraufwand bezüglich Design, Programmierung und Synchronisation der Benutzungsoberfläche, da die Medien der verschiedenen Sprachen jederzeit austauschbar sein müssen. Des Weiteren muss ein Knopf für den Sprachwechsel in die Benutzungsoberfläche integriert werden.

7.4.1.8. Konzipiere die Umsetzung der Lerneranalyse

Für die Lerneranalyse muss die *Architektur*[Artefakt] des virtuellen Labors die Protokollierung von Daten des Lerners ermöglichen, das heißt es müssen Lernerdaten erhoben, gespeichert,

verändert und ausgelesen werden können. Zur Umsetzung der Lerneranalyse bietet sich daher die Verwendung einer Datenbank an. Dazu sind in der Datenbank entsprechende Tabellen und Sichten gemäß der in der Aktivität *Konzipiere die Lerneranalyse*[Aktivität] (siehe Abschnitt 7.3.3.4) festgelegten Protokoll-Struktur anzulegen. Des Weiteren wird die Datenbank über eine geeignete Schnittstelle an das virtuelle Labor angebunden.

7.4.2. Verfeinere die Architektur

Die in Kapitel 7.4.1 festgelegte *Architektur*[Artefakt] des virtuellen Labors wird in dieser Aktivität von den *Designern/Entwicklern*[Rolle] und den *Simulationsspezialisten*[Rolle] verfeinert. Ziel der Verfeinerung der *Architektur*[Artefakt] ist es, den natürlichen Übergang von der objektorientierten Analyse zum objektorientierten Entwurf bereit zu stellen (Kruchten, 2000). Dazu wird zum einen das intelligente Tutorsystem betrachtet. Dieses wird insofern verfeinert, als dass die einzelnen Module, das heißt das Lerner-, Lehrer- und Expertenmodul, entworfen werden und deren gegenseitiges Zusammenwirken festgelegt wird. Problematisch ist dabei allerdings, dass es zwar eine allgemeine Architektur für intelligente Tutorsysteme (siehe Abbildung 2.1) gibt, diese jedoch nicht die Art der Interaktion und Kooperation der einzelnen Module beschreibt. Außerdem existieren für die Implementierung der Module wiederum jeweils eine Vielzahl von Möglichkeiten (siehe Witschital, 1990, Seite 32 bis 61). Bei der Verfeinerung der *Architektur*[Artefakt] werden des Weiteren die einzelnen Komponenten des Simulators identifiziert. So können zum Beispiel bestimmte Reaktionsabläufe oder Interaktionen mit den Behältern und Laborgeräten identifiziert und zu einer Klasse zusammengefasst werden. Ist für das virtuelle Labor eine Client/Server-Verteilung vorgesehen, so ist in dieser Aktivität die verwendete Kommunikationstechnik zwischen Client und Server festzulegen. Außerdem ist bei Verwendung einer Datenbank oder eines Expertensystems die konkrete Anbindung an das virtuelle Labor zu bestimmen. Gegebenenfalls lassen sich auch noch weitere Möglichkeiten identifizieren, um Entwurfsmuster anzuwenden.

7.4.3. Analysiere das Verhalten

In dieser Aktivität werden die Anwendungsfälle des *Anwendungsfall-Modells*[Artefakt] von den *Designern/Entwicklern*[Rolle] und den *Simulationsspezialisten*[Rolle] analysiert. Es werden also die Anwendungsfälle zur Auswahl und Durchführung der Versuche betrachtet und die möglichen Versuchsabläufe und Fehlerfälle durchlaufen. Auch werden die Anwendungsfälle zum Abruf von Hintergrundwissen und zum dynamischen Laden und Speichern von Versuchen analysiert. Ziel der Analyse ist es, Elemente zu identifizieren, auf deren Basis der objektorientierte Entwurf durchgeführt werden kann.

7.4.4. Entwerfe die Komponenten

Auf der Basis des *OOA-Modells*[Artefakt], der *Architektur*[Artefakt] und dem Metaobjektmodell für virtuelle Labore wird der Entwurf der Komponenten des virtuellen Labors durchgeführt. Die

Komponenten virtueller Labore sind die Arbeitsflächen, Versuchskomponenten und deren gegenseitiges Zusammenwirken im Simulationsmodell. Weitere Beispiele für Komponenten sind das Lerner-, Lehrer- und Expertenmodul des intelligenten Tutorsystems, sowie die weiteren Bestandteile des virtuellen Labors, wie die Transportleiste und das Anleitungsfenster.

Generell gilt für den Entwurf der Komponenten des virtuellen Labors, dass entsprechende *Model*-, *View*- und *Controller*-Klassen zu entwerfen sind. Dazu werden entsprechende Klassendiagramme erstellt. Ergebnis dieser Aktivität ist das *OOD-Modell*[Artefakt], das das virtuelle Labor mittels Klassen und Klassengruppen beziehungsweise Paketen beschreibt (siehe dazu den OOD-Architekturentwurf in Balzert, 2000b, Seite 936 ff.). Diese Aktivität ist Gegenstand des objektorientierten Entwurfs und wird von den *Designern/Entwicklern*[Rolle] und den *Simulationsspezialisten*[Rolle] durchgeführt.

7.4.5. Entwerfe die Echtzeit-Komponenten (optional)

Diese Aktivität ist optional und wird von den *Designern/Entwicklern*[Rolle] oder den *Simulationsspezialisten*[Rolle] durchgeführt. Echtzeit-Komponenten die in virtuellen Laboren eingebunden werden sollen, sind zum Beispiel ein in Echtzeit berechnetes Simulationsmodell. Auch kann die Anbindung von externen Geräten an das virtuelle Labor, wie beispielsweise ein externes Messinstrument, die Einbindung von Echtzeit-Komponenten erfordern. Dazu wird die Schnittstelle sowie das Protokoll zu dem externen Gerät entworfen und in der Aktivität *Implementiere die Komponenten*[Aktivität] (siehe Kapitel 7.6.3) implementiert.

7.4.6. Entwerfe die Datenbank (optional)

Diese optionale Aktivität wird von dem *Datenbank-Entwickler*[Rolle] durchgeführt. Dieser entscheidet, welches Datenbankmodell verwendet werden soll. Des Weiteren wird die Schnittstelle zu den Komponenten des virtuellen Labors entworfen. Datenbanken können in virtuellen Laboren zum Beispiel zur Speicherung der Medien, das heißt der Texte, Graphiken, 3D-Modelle, Animationen, Audio- und Videosequenzen verwendet werden. Auch kann der aktuelle Zustand der Versuchsdurchführung in der Datenbank gespeichert werden. Dazu sind das Simulationsmodell und das Lernermodell sowie jeweils alle dazugehörenden Objekte in der Datenbank abzulegen. Nach (Yass, 2000, Seite 152) kann durch den Einsatz einer Datenbank die Speicherung von Textinhalten die Entwicklung eines Multimedia-System beschleunigt und flexibilisiert werden. Der hauptsächliche Einsatz einer Datenbank in virtuellen Laboren wird jedoch die Protokollierung von Lernerdaten sein (siehe dazu *Konzipiere die Umsetzung der Lerneranalyse*[Aktivität] in Abschnitt 7.4.1.8). Im Rahmen dieser Arbeit wird nicht näher auf die Wahl und den Entwurf einer geeigneten Datenbank und die Anbindung an das virtuelle Labor eingegangen. Statt dessen sei hier exemplarisch auf (Heuer und Saake, 2000) verwiesen.

7.4.7. Entwerfe das Expertensystem (optional)

Diese Aktivität ist optional und wird von den *Datenbank-Entwicklern*[Rolle] und den *Fachexperten*[Rolle] durchgeführt. Es ist zu entscheiden, welches Expertensystem eingesetzt werden soll und wie dieses an das virtuelle Labor angebunden wird. Dabei kann das Expertensystem dem Lerner beispielsweise Ratschläge zur Versuchsdurchführung geben, oder aber auch das Simulationsmodell einer stochastischen Simulation (siehe Kapitel 2.6) ergänzen oder sogar ersetzen.

Im Rahmen dieser Arbeit sei noch erwähnt, dass man bei Expertensystemen grundsätzlich zwischen zwei Kategorien von Expertenmodellen unterscheidet. Bei dem *Glass-Box*-Expertenmodell können die Entscheidungsfindungen des Expertensystems von außen eingesehen und die einzelnen Schritte zum Ergebnis nachvollzogen werden. Dagegen werden bei einem *Black-Box*-Expertenmodell zwar Antworten auf die Anfragen an das Expertensystem generiert, der Prozess der Entscheidungsfindung bleibt jedoch verborgen (siehe Witschital, 1990, Seite 35 f.).

7.5. Medienproduktion

In dem Workflow der *Medienproduktion*[Workflow] werden von den *Medienspezialisten*[Rolle] sämtliche multimedialen Elemente (*Medien*) erstellt, die in der aktuellen Iteration für das virtuelle Labor benötigt werden. Das sind beispielsweise realitätsgetreue Abbilder der eingesetzten Laborgeräte in Form von 3D-Modellen oder Animationen für die Erklärung von Reaktionsprozessen in einem Versuch. Der konkrete Ablauf der *Medienproduktion*[Workflow] ist in Abbildung 7.20 dargestellt.

In der *Medienproduktion*[Workflow] besteht zunächst die Möglichkeit das allgemeine Layout und die Gestaltung der Benutzungoberfläche zu entwerfen. Diese kann in jeder Iteration überarbeitet und verändert werden. Des Weiteren wird in der *Medienproduktion*[Workflow] eine Analyse der benötigten und der vorhandenen Medien vorgenommen. Anschließend wird die eigentliche *Medienproduktion*[Workflow] geplant. Bei der Durchführung der *Medienproduktion*[Workflow] teilt sich der Workflow in vier parallele Pfade. Jeder Pfad betrachtet eine mögliche Variante der Herstellung der Medien. Die Varianten sind die Digitalisierung von Medien, die Überarbeitung vorhandener Medien, die Herstellung neuer Medien sowie die Beschaffung von Medien (vergleiche dazu Multimedia-Editing in Kerres, 1998, Seite 338 f.).

Die Aktivitäten der *Medienproduktion*[Workflow] werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Erstelle das Gesamtdesign*[Aktivität] (siehe Kapitel 7.5.1),
- *Ermittle benötigte Medien*[Aktivität] (siehe Kapitel 7.5.2),
- *Analysiere vorhandene Medien*[Aktivität] (siehe Kapitel 7.5.3),
- *Strukturiere die Medienproduktion*[Aktivität] (siehe Kapitel 7.5.4),

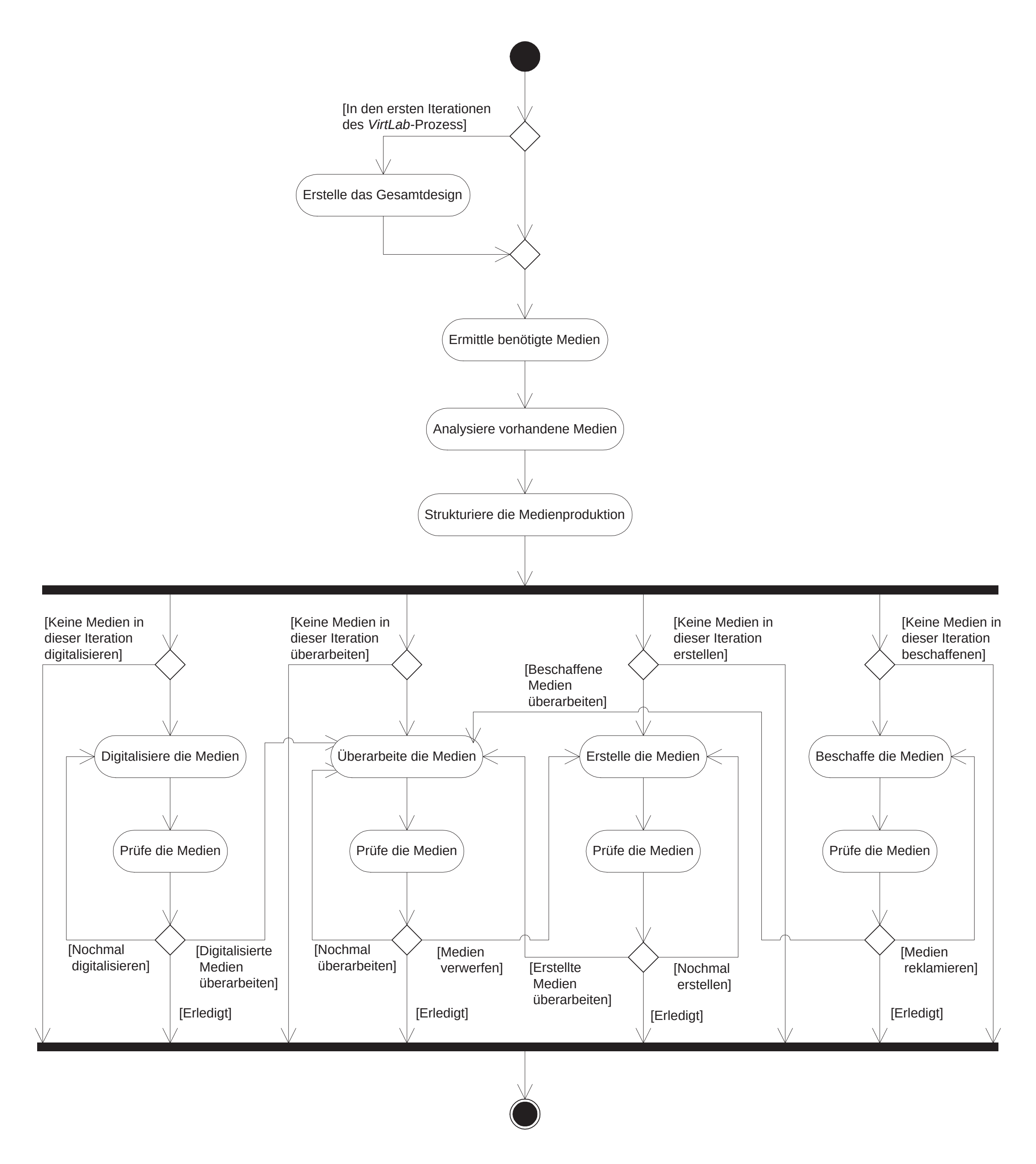


Abbildung 7.20.: Der Workflow für die Medienproduktion

- *Digitalisiere die Medien*^Aktivität (siehe Kapitel 7.5.5),
- *Überarbeite die Medien*^Aktivität (siehe Kapitel 7.5.6),
- *Erstelle die Medien*^Aktivität (siehe Kapitel 7.5.7),
- *Beschaffe die Medien*^Aktivität (siehe Kapitel 7.5.8) und

- *Prüfe die Medien*[Aktivität] (siehe Kapitel 7.5.9).

7.5.1. Erstelle das Gesamtdesign

In den ersten Iterationen des *VirtLab*–Prozesses wird von den *Medienspezialisten*[Rolle] das allgemeine Layout und die Gestaltung der Benutzungsoberfläche entworfen. Des Weiteren wird die Farbtiefe und die Bildschirmauflösung des virtuellen Labors festgelegt. In den weiteren Aktivitäten der *Medienproduktion*[Workflow] werden dann die Medien für einen entsprechenden Prototypen (*Oberflächenprototyp*) erstellt. Die grundsätzlichen Gestaltungsempfehlungen für die Erstellung von Benutzungsoberflächen bleiben für virtuelle Labore bestehen. Die wichtigste Vorgabe für die Gestaltung der Benutzungsoberfläche eines virtuellen Labors ist, dass die Arbeitsflächen, Laborgeräte, Behälter und Substanzen möglichst realitätsnah darzustellen sind. Nur durch eine entsprechende Realitätsnähe der Benutzungsoberfläche ist der Transfer der im virtuellen Labor erlernten Handlungsabläufe auf die Realität gewährleistet (nach Schult, 1994; Heuten, 2001). Diese wesentliche Anforderung an die Gestaltung der Benutzungsoberfläche unterscheidet virtuelle Labore von allgemeinen multimedialen Lehr- und Lernsystemen. Letztere dürfen auch andere künstlerische Ansprüche und Anforderungen erheben, wie zum Beispiel das Design der Benutzungsoberfläche in Form eines Zeichentrickfilms. Bezüglich des Layouts, das heißt der Bildschirmaufteilung ist zu entscheiden, an welcher Stelle sich das Anleitungsfenster, die Transportleiste, die allgemeinen Navigationselemente und die Arbeitsfläche befinden sollen. Sowohl die Bildschirmaufteilung als auch das Design der Benutzungsoberfläche des virtuellen Labors kann an *GenLab* angelehnt werden. Werden vom *Auftraggeber*[Rolle] Vorgaben des *Corporate Designs* definiert, so sind diese bei der Gestaltung der Benutzungsoberfläche einzuhalten. Außerdem besteht die Möglichkeit das Layout und die Gestaltung der Benutzungsoberfläche in den nachfolgenden Iterationen zu überarbeiten und zu korrigieren.

7.5.2. Ermittle benötigte Medien

In diesem Teil der *Medienproduktion*[Workflow] werden zunächst alle für die aktuelle Iteration benötigten Medien ermittelt. Dazu treffen sich zum Iterationsauftakt alle an der Entwicklung des virtuellen Labors beteiligten *Projektmitarbeiter*[Rolle]. Allgemein werden Medien zur visuellen und auditiven Darstellung des virtuellen Labors benötigt. Das sind zum Beispiel die Abbilder der Arbeitsflächen, Laborgeräte, Behälter und Substanzen. Aber auch zur Darstellung des Hintergrundwissens, der Transportleiste, des Anleitungsfensters und der Navigation im virtuellen Labor werden Medien benötigt. Des Weiteren ergeben sich aus den Vorgaben des *didaktischen Konzeptes*[Artefakt] bezüglich des Interaktionsdesigns weitere Medien, wie zum Beispiel für eine Dialogbox zur Auswahl der benötigten Substanzen. Es werden Typen, Anzahl sowie weitere Merkmale der benötigten Medien, wie beispielsweise Größe und Auflösung, ermittelt und in der *Liste der benötigten Medien*[Artefakt] festgehalten. Zu jedem Eintrag in der Auflistung wird vermerkt, ob das Medium *vorhanden* oder *nicht vorhanden*

ist. Zudem erhält jeder Eintrag ein Attribut, ob das Medium *digitalisiert*, *überarbeitet*, *erstellt* oder *beschaffen* werden soll.

Prinzipiell sind alle Medientypen, die in einem beliebigen Multimedia-System nach Kapitel 2.5 auftreten können, auch bei der Erstellung von virtuellen Laboren zu berücksichtigen. Bezüglich virtueller Labore lassen sich außerdem spezielle Medientypen identifizieren beziehungsweise allgemeine Medientypen präzisieren. Demnach lassen sich für virtuelle Labore folgende Medientypen unterscheiden:

- *Graphik*: Eine Graphik ist allgemein ein zweidimensionales *Bitmap*, dass auf dem Bildschirm dargestellt werden kann. Graphiken können beliebige Symbole, Knöpfe, Regler, Navigationselemente und Hintergundbilder sein, die in das virtuelle Labor eingefügt werden. Aus Graphiken setzt sich zum Beispiel die Darstellung der Transportleiste und die Laborübersicht zusammen. Sollen die Graphiken später animiert werden, so können diese auch als Vektorgraphiken abgespeichert werden (Sawhney, 1995, Seite 47 f.).

- *Animation*: Eine Animation ist die zeitkontinuierliche Änderung der Attribute eines oder mehrerer Medien. Dabei beschränkt sich die Animation eines Mediums, wie zum Beispiel eine Graphik, nicht nur auf die Veränderung der Position (*Bewegung*) auf dem Bildschirm. Neben diesem Attribut lässt sich das Medium Graphik über weitere Attribute wie beispielsweise Farbe, Größe, Orientierung und Oberflächenstruktur animieren. Auch nicht sichtbare Medien können animiert werden. So lassen sich Audiosequenzen in Lautstärke, Frequenz und Klang verändern. Generell gilt, dass alle Attribute eines Mediums für eine Animation verwendet werden können (vergleiche Boles, 1998). Animationen werden im virtuellen Labor zum Beispiel zur Veranschaulichung von Versuchsabläufen und Reaktionsprozessen eingesetzt.

- *Text*: In der Regel wird ein Großteil des theoretischen Hintergrundwissens der Domäne des virtuellen Labors als Text vorliegen. Aber auch die Beschreibung der Versuchskomponenten oder ein *Tooltip* stellt einen Text dar.

- *3D-Modell*: Allgemein ist ein 3D-Modell eine funktionierende mathematische Konstruktion eines real existierenden Objektes und wird von einem *3D–Modellierungs-spezialisten*[Rolle] erstellt (vergleiche Heuten, 2001, Seite 106 bis 110).[12] Im virtuellen Labor ist ein 3D-Modell zum Beispiel eine dreidimensionale Nachbildung eines realen Laborgerätes, Behälters oder einer Arbeitsfläche. Ein 3D-Modell beinhaltet im Idealfall noch zusätzliche Informationen über die Bewegungsabläufe der realen Laborgeräte[13]. Sind Informationen über die Bewegungsabläufe nicht im 3D-Modell enthalten, so kann das 3D-Modell über die einzelnen Bestandteile (vordefinierte Primitive) des zu modellierenden Objektes erstellt werden (vergleiche Heuten, 2001, Seite 108 bis 110).

[12] Zum Beispiel mit dem Werkzeug *Cinema 4D* von MAXON Computer, Inc. (siehe: http://www.maxoncomputer.com/)

[13] Beispiel: Das 3D-Modell einer Mikrowelle in *GenLab* besteht nicht nur aus einem virtuellen Abbild des Gerätes, sondern beinhaltet noch zusätzliche Informationen über den Öffnungsmechanismus der Tür, den Drehreglern zur Einstellung von Kochzeit und Leistung sowie den Knopf zum Öffnen der Tür.

Des Weiteren kann das virtuelle Labor selbst, das heißt die Räume in denen sich die Arbeitsflächen befinden, mit Hilfe eines 3D-Modells realisiert werden, wie zum Beispiel in *GenLab* (siehe Hasler und Schlattmann, 2001, Seite 9).

- *2D-Ansicht*: Eine 2D-Ansicht einer Versuchskomponente besteht aus einer *Vordergrundgraphik*, *Hintergrundgraphik*, *Standfläche*, *Miniaturansicht* und gegebenenfalls einer weiteren *dynamischen Komponente* (vergleiche Heuten, 2001, Seite 3 f.). Die Vordergrundgraphik stellt ein realistisches Abbild des Laborgerätes dar und setzt sich aus ein bis n Teilgraphiken zusammen. Für die interaktiven Teile einer 2D-Geräteansicht existieren sensitive Bereiche, die auf Eingaben der Computer-Maus reagieren. Diese stellen im gewissen Sinne eine Hintergrundgraphik der 2D-Ansicht dar und werden durch sogenannte *Bitmap-Masken* (oder kurz *Masken*) realisiert. Wie die Vordergrundgraphik, so setzt sich auch die Hintergrundgraphik aus ein bis n Teilmasken zusammen. Die Standfläche einer 2D-Ansicht ist die Projektion des 3D-Modells auf die Arbeitsebene. Diese wird in der 2D-Darstellung verwendet, um in Verbindung mit der Z-Sortierung und der Kollisionserkennung (siehe *Konzipiere Mechanismen der explorativen Lehr- und Lernumgebung*^Aktivität in Abschnitt 7.4.1.4) eine gewisse Tiefenwirkung (*Dreidimensionalität*) zu erzielen. *Miniaturansichten* werden für die Transportleiste des virtuellen Labors benötigt. Es besteht zudem die Möglichkeit die an sich statischen 2D-Ansichten um eine dynamische Komponente zu ergänzen.[14] Ein Beispiel einer 2D-Ansicht ist in der Abbildung 7.21 dargestellt und aus *GenLab* entnommen. Auf die Verwendung der Hintergrundgraphiken kann verzichtet werden, falls für die Implementierung beispielsweise das Autorensystem *Director* zur Verfügung steht. In diesem Fall werden Vordergrundgraphiken anstelle von Hintergrundgraphiken zur Realisierung der Masken der 2D-Ansicht verwendet (wie zum Beispiel in *GenLab* geschehen).

- *2D-Arbeitsfläche*: Eine 2D-Arbeitsfläche ist eine einfache Graphik. Diese wird beispielsweise aus einem 3D-Modell der Arbeitsfläche erstellt und als Hintergrundgraphik im virtuellen Labor verwendet.

- *3D-Ansicht*: Eine 3D-Ansicht ist die direkte Darstellung eines 3D-Modells einer Versuchskomponente des virtuellen Labors in der 3D-Darstellung.

- *3D-Arbeitsfläche*: Eine 3D-Arbeitsfläche ist die direkte Darstellung eines 3D-Modells einer Arbeitsfläche im virtuellen Labor in der 3D-Darstellung.

- *Ansicht von allen Seiten*: Die Ansicht von allen Seiten zeigt Gegenstände des virtuellen Labors losgelöst von der Umgebung frei drehbar im dreidimensionalen Raum und wird auf der Grundlage von 3D-Modellen erstellt. Alle Aspekte des Gegenstandes können somit dem Lerner gezeigt werden (vergleiche Yass, 2000, Seite 74). Eine Ansicht von allen Seiten wird in virtuellen Laboren typischerweise für Laborgeräte, Behälter und Zubehör realisiert. Wird für die Benutzungsoberfläche des virtuellen Labors die

[14] Diese stehen zum Beispiel beim Autorensystem *Director* zur Verfügung und werden dort als *Vectorshapes* bezeichnet.



Abbildung 7.21.: 2D-Ansicht am Beispiel eines Eisbehälters (aus *GenLab*)

2D-Darstellung gewählt, so erfolgt die Ansicht von allen Seiten beispielsweise mit Hilfe eines interaktiven Videos[15]. Bei virtuellen Laboren in der 3D-Darstellung kann die Ansicht von allen Seiten eines 3D-Modells direkt mit Hilfe der eingesetzten *3D-Darstellungskomponente* erfolgen und entspricht der 3D-Ansicht (siehe oben).

- *Audiosequenzen*: Mit Hilfe von Audiosequenzen können Arbeitsflächen und Versuchskomponenten des virtuellen Labors vertont werden. In diesem Fall sind Audiosequenzen die verschiedenen Töne und Geräusche, die von den Arbeitsflächen und Versuchskomponenten erzeugt werden, wie zum Beispiel das Geräusch einer Abzugsanlage oder das Klicken eines Schalters. Dabei können die verwendeten Töne und Geräusche zuvor in einem realen Labor aufgezeichnet worden sein. Audiosequenzen können auch als akkustische Signale des intelligenten Tutorsystems zur Unterstützung der Benutzungsoberfläche verwendet werden. Außerdem werden auch Sprache und Musik als Audiosequenzen verstanden. Sprache kann zur Vertonung von Animationen und zur Erläuterung der Bedienung des virtuellen Labors verwendet werden. Musik findet in virtuellen Laboren kaum Verwendung und wird zum Beispiel zur Begrüßung des Lerners nach dem Start des virtuellen Labors eingesetzt. Werden Audiosequenzen für das virtuelle Labor verwendet, so ist es wichtig, dass auch bei *Stumm-Schaltung* (das heißt ohne Audiosequenzen) das virtuelle Labor für den Lerner verständlich ist (Merx, 1999, Seite 127).

- *Videosequenzen*: Videosequenzen sind zum Beispiel Aufzeichnungen von Vorgängen in einem realen Labor, die sonst nur schwer vermittelbar sind, das heißt ohne die eine Übertragung der erlernten Handlungsabläufe im virtuellen Labor auf die Realität nur

[15] Zum Beispiel *QuickTime* von *Apple* (siehe: http://www.apple.com/quicktime/)

schwer möglich ist. Dies ist zum Beispiel beim Sezieren von einem Frosch der Fall. Oft finden Videosequenzen daher Verwendung in multimedialen Lehr- und Lernsystemen die zur Ersetzung von Tierversuchen dienen, wie zum Beispiel in dem virtuellen Labor *SimNerv* aus der Reihe *Virtual Physiology* des *Georg Thieme Verlags* (siehe Virtual Physiology - Internetseiten, 2001). Die Vertonung einer Videosequenz geschieht mit Hilfe einer Audiosequenz, die entweder fester Bestandteil der Videosequenz sein kann, oder getrennt davon vorliegt.

- *Wegwerf-Medium*: Das Wegwerf-Medium ist ein Metamedientyp und kann von einem beliebigen der zuvor genannten Medientypen sein. Wegwerf-Medien werden für Testzwecke erstellt oder wenn das eigentliche Medium noch nicht verfügbar ist. Ein Wegwerf-Medium ist zum Beispiel ein vorläufiges Abbild eines virtuellen Laborgerätes, das noch nicht der gewünschten Farbgebung oder Modellierungsgenauigkeit entspricht. Ob in der aktuellen Iteration tatsächlich Wegwerfmedien zu erstellen sind, wird während der *Implementierung*[Workflow] in der Aktivität *Integriere die Medien*[Aktivität] (siehe Kapitel 7.6.4) entschieden.

Die Auswahl der benötigten Medien erfolgt immer als Mittel eines pädagogischen Zwecks, das heißt die Medien werden stets didaktisch sinnvoll eingesetzt und verkommen nicht zum Selbstzweck (vergleiche Hesse u. a., 1997, Seite 265). Statt Reizüberflutung sollte bei dem Design des virtuellen Labors also auf überflüssige visuelle und auditive Effekte verzichtet werden (Merx, 1999, Seite 140).

Außerdem ist bei einem mehrsprachigen virtuellen Labor die Benutzungsoberfläche für die jeweilige Sprachversion anzupassen. Dabei sind nach (Yass, 2000, Seite 259) die folgenden Bestandteile des virtuellen Labors zu berücksichtigen: Layout der Benutzungsoberfläche, Texte, Graphiken, Audiosequenzen, wie zum Beispiel gesprochene Texte und Videosequenzen mit Sprachausgabe.

7.5.3. Analysiere vorhandene Medien

Nachdem die benötigten Medien ermittelt wurden, sind als nächstes alle bereits vorhandenen, das heißt eigenen oder aus anderen Quellen[16] verfügbaren Medien für eine Wiederverwendung zu prüfen. Dazu wird von den *Medienspezialisten*[Rolle] die *Liste der benötigten Medien*[Artefakt] mit den bereits vorhandenen Medien abgeglichen und die Verwendbarkeit der Medien für die nächste Iteration geprüft. Für jedes physikalische Medium wird entschieden, ob es digitalisiert wird. Für jedes vorhandene Medium wird entschieden, ob es wiederverwendet, überarbeitet oder nicht verwendet wird. Für jedes nicht vorhandene Medium wird entschieden, ob es erstellt oder beschaffen wird (*Make-Or-Buy-Entscheidung*). Werden Medien aus anderen Quellen verwendet, so müssen eventuell vorliegende Urheberrechte geprüft und gegebenenfalls Nutzungslizenzen erworben werden (siehe Heinrich und Schifman, 2000, Seite 150 f.). Die Ergebnisse dieser Aktivität werden in einer aktualisierten Version der *Liste der benötigten Medien*[Artefakt] festgehalten.

[16] Zum Beispiel Bücher, Ressourcen im Internet oder spezielle Multimedia-CDs

7.5.4. Strukturiere die Medienproduktion

In dieser Aktivität wird die eigentliche *Medienproduktion*[Workflow] geplant. Dazu wird von den *Medienspezialisten*[Rolle] auf der Grundlage der *Liste der benötigten Medien*[Artefakt] eine Zeit- und Kostenschätzung der *Medienproduktion*[Workflow] erstellt (siehe dazu *Schätze Umfang und Risiko des Projekts*[Aktivität] in Kapitel 7.1.2). Diese wird vom *Projektleiter*[Rolle] mit der Zeit- und Kostenschätzung im *Gesamtentwicklungsplan*[Artefakt] verglichen. Ziel ist es, die *Medienproduktion*[Workflow] möglichst effizient zu halten, damit die Kosten nicht zu hoch werden. Hängen die Medien voneinander ab, so wird festgelegt, in welcher Reihenfolge diese zu produzieren sind. Sind die Medien unabhängig voneinander, so kann die Digitalisierung, Erstellung, Überarbeitung und Beschaffung der Medien parallel erfolgen (Sawhney, 1995, Seite 47). Des Weiteren wird festgelegt, welches Medium von welchem *Medienspezialisten*[Rolle] produziert wird und welche Werkzeuge dabei verwendet werden. Nach Bestätigung der Planung durch den *Projektleiter*[Rolle] kann die eigentliche *Medienproduktion*[Workflow] beginnen.

7.5.5. Digitalisiere die Medien

In dieser Aktivität werden die physikalischen Medien digitalisiert und in elektronisch zugreifbarer Form gespeichert. Diese Aktivität wird von einem *Digitalisierer*[Rolle] durchgeführt. In (Steinmetz, 1999, Seite 838 bis 840) wird in diesem Zusammenhang auch von der *Medienaufbereitung* gesprochen. Es wird dabei zwischen der physikalischen (*realen*) Welt und der elektronischen (*virtuellen*) Welt unterschieden. Um Medien der physikalischen Welt in die elektronische Welt zu transformieren, bedarf es spezieller Geräte (*Scanner*). So werden *Flachbettscanner* für die Digitalisierung von gedruckten Texten, Graphiken und Photos verwendet. Des Weiteren werden *3D-Scanner* zur Erstellung von 3D-Modellen anhand real existierender Objekte eingesetzt. Dabei ist die Digitalisierung von 3D-Modellen nicht zu verwechseln mit der Erstellung von 3D-Modellen durch die *3D–Modellierungsspezialisten*[Rolle].

7.5.6. Überarbeite die Medien

Die Überarbeitung der vorhandenen Medien wird in diesem Teil der *Medienproduktion*[Workflow] durchgeführt und betrifft prinzipiell alle Medientypen. Die zur Überarbeitung der Medien durchzuführenden Aktivitäten hängen von dem jeweiligen Medientyp ab. Es sind beispielsweise 3D-Modelle von den *3D–Modellierungsspezialisten*[Rolle] anzupassen oder einzelne Graphiken von einem *Graphik-Designer*[Rolle] mit einem geeigneten Bildbearbeitungsprogramm zu korrigieren. Die Audio- und Videosequenzen werden von den *Audio- und Videospezialisten*[Rolle] überarbeitet. Dabei kommen ebenfalls spezielle Werkzeuge zum Einsatz.

Die Konvertierung von Medien stellt ebenfalls eine Überarbeitung dar, zum Beispiel wenn die Qualität der vorliegenden Medien zu reduzieren ist. Dabei erfolgt die Konvertierung immer

innerhalb desselben Medientyps. So kann Zum Beispiel eine Videosequenz, die in einer sehr hohen Qualität, das heißt in sehr hoher Farbtiefe und Bildauflösung vorliegt, in eine Videosequenz von geringerer Qualität umgewandelt werden. Ändert sich der Medientyp, zum Beispiel wenn aus einem 3D-Modell eines Laborgerätes zweidimensionale Graphiken erzeugt werden, so wird von einer Medienerstellung (siehe nachfolgendes Kapitel 7.5.7) gesprochen.

7.5.7. Erstelle die Medien

In diesem Teil der *Medienproduktion*^Workflow werden die benötigten Medien des virtuellen Labors hergestellt. Die dabei durchzuführenden Aktivitäten bei der Erstellung eines neuen Mediums sind von dem jeweiligen Medientyp abhängig. Die Medienerstellung betrifft nicht nur die technische Konstruktion mit Hilfe von Werkzeugen, sondern auch das Design der Medien (siehe auch *Erstelle das Gesamtdesign*^Aktivität in Kapitel 7.5.1).

Zur Erstellung der Medien ist unter Umständen ein mehrstufiger Entwicklungsprozess nötig. Beispiel eines solchen Entwicklungsprozesses ist die Erstellung realitätsgetreuer Abbilder von Laborgeräten und Behältern für die 2D-Darstellung mittels 3D-Modelle. Dabei können für die Erstellung der 3D-Modelle von Versuchskomponenten verschiedene Quellen genutzt werden. So können zum Beispiel reale Laborgeräte von den Geräteherstellern für die Medienproduktion zur Verfügung gestellt werden. Diese werden von den *3D–Modellierungsspezialisten*^Rolle im Detail analysiert und modelliert. Es ist aber auch denkbar, dass die realen Laborgeräte direkt digitalisiert werden (siehe dazu *Digitalisiere die Medien*^Aktivität in Kapitel 7.5.5). Eventuell werden auch die CAD-Zeichnungen (*Computer Aided Design*) der Laborgeräte von den Geräteherstellern zur Verfügung gestellt. Eine andere Quelle für die Erstellung der 3D-Modelle sind die Kataloge der Gerätehersteller und andere (Lehr-)Bücher zu der Domäne des virtuellen Labors. Aus diesen können beispielsweise auch Photos, technische Daten und eine Beschreibung der Laborgeräte entnommen werden. Für alle verwendeten Quellen sind die Nutzungsrechte mit den Geräteherstellern und Verlagen zu klären und entsprechende Lizenzen zu erwerben (siehe dazu Heinrich und Schifman, 2000, Seite 150 f.). Es können aber auch eigene Photos und Videoaufzeichnungen von den Versuchskomponenten erstellt werden. Diese eignen sich dann nicht nur als Grundlage für die Erstellung der 3D-Modelle, sondern können zum Beispiel auch zur Darstellung des entsprechenden Hintergrundwissens verwendet werden (siehe *Spezifiziere das Hintergrundwissen zur Lerneinheit* in Kapitel 7.3.5).

Bei der Erstellung von 3D-Modellen legen die *3D–Modellierungsspezialisten*^Rolle in Absprache mit dem *Fachdidaktiker*^Rolle fest, welche Elemente des zu modellierenden Laborgerätes wichtig sind und welche unwichtig. Wichtige Elemente, wie notwendige Schalter, Knöpfe und Regler zur Bedienung eines Laborgerätes, müssen detailliert nachgebildet werden, da sie die Funktionalität des späteren virtuellen Laborgerätes darstellen. Diese müssen in der Regel beweglich sein (außer zum Beispiel Sensor-Tasten) und sind bei der Implementierung des virtuellen Laborgerätes entsprechend zu berücksichtigen. Unwichtige Elemente können weniger genau modelliert oder sogar weggelassen werden, falls diese für die Versuchsdurchführung nicht benötigt werden. Das Laborgerät wird in diesem Fall also nur

zum Teil modelliert und entspricht gerade der Modellierungsgenauigkeit, die für das virtuelle Labor benötigt wird. Dadurch werden zum einen die Herstellungskosten des 3D-Modells reduziert. Zum anderen wird durch ein einfacheres Modell des Laborgerätes der Lerneffekt erhöht, da sich der Lerner auf die für ihn wichtigen Geräteeigenschaften konzentrieren kann und sich nicht mit der eigentlichen Komplexität des Laborgerätes und den für die Versuchsdurchführung unwichtigen Eigenschaften beschäftigen muss (vergleiche Kronberg, 1998, Seite 30). Bei der Erstellung von 3D-Modellen sind des Weiteren die Probleme der Virtualisierung zu beachten. Diese werden in der Aktivität *Spezifiziere Aufbau und Ablauf des Versuchs bzw. der Fertigkeit*^Aktivität (siehe Abschnitt 7.3.4.1) beschrieben.

Für die Erstellung von weiteren Medien aus den 3D-Modellen wird in (Heuten, 2001) zwischen der 2D-Darstellung und der 3D-Darstellung unterschieden (siehe *Definiere das virtuelle Labor*^Aktivität in Kapitel 7.2.3). Ausgangsbasis beider Darstellungsarten sind die 3D-Modelle, aus denen die Medien für die 2D- oder 3D-Darstellung im virtuellen Labor erzeugt werden. In (Heuten, 2001) wird in diesem Zusammenhang von den *Repräsentationen* der 3D-Modelle im virtuellen Labor gesprochen. Die möglichen *Repräsentationstypen* sind für die 2D- und 3D-Darstellung identisch, allerdings ist die zugrunde liegende Technologie, das heißt 2D-Bitmaps oder 3D-Modelle, verschieden (vergleiche Heuten, 2001, Seite 5 und 7).

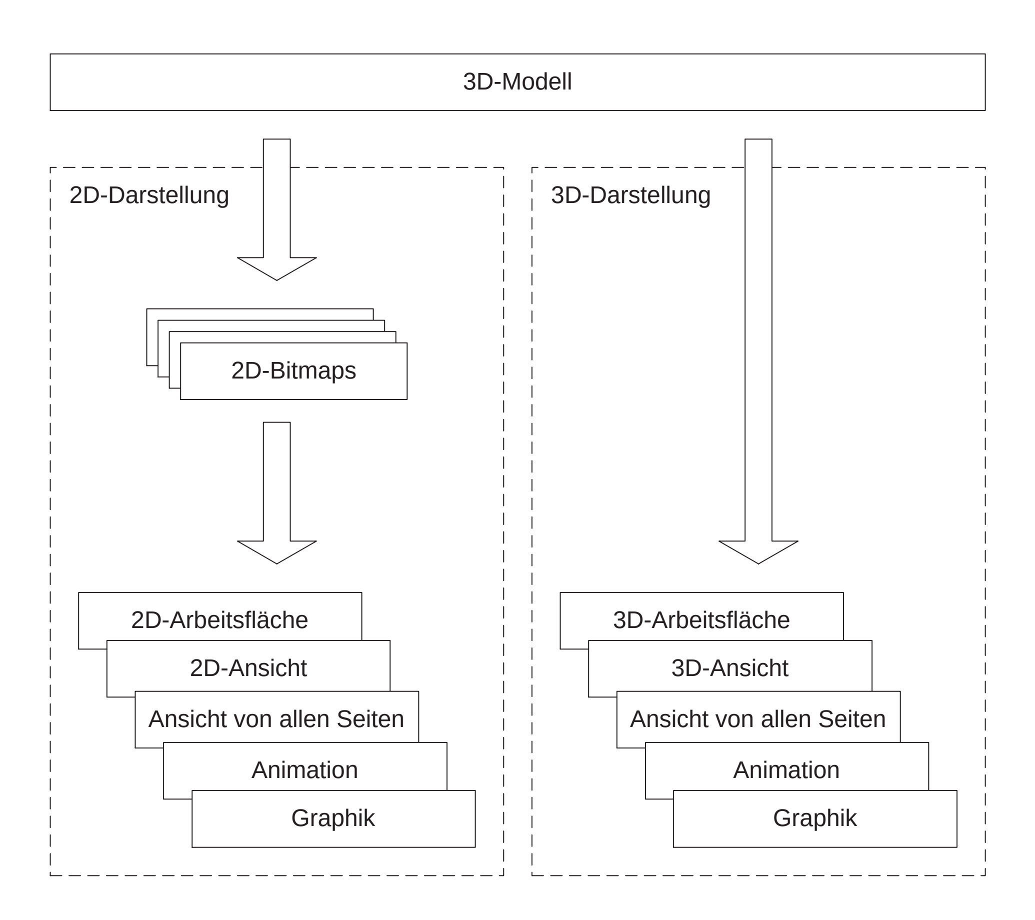


Abbildung 7.22.: Darstellungsarten von 3D-Modellen in virtuellen Laboren und ihre Repräsentationen (nach Heuten, 2001)

Wird für das virtuelle Labor die 2D-Darstellung gewählt, so werden *2D-Bitmaps* zur Erstellung der Repräsentationen der 3D-Modelle verwendet (siehe Abbildung 7.22). Bei der 3D-Darstellung können die 3D-Modelle direkt verwendet werden. Zur Erstellung von Animationen für die 2D-Darstellung eignen sich besonders 3D-Modelle, die mit Zeitinformation behaftet sind (sogenannte *4D-Modelle*). Aus dem 4D-Modell werden in bestimmten Zeitabständen einzelne Graphiken erzeugt und anschließend zu einer Animation zusammengefügt. Für den Druck auf Papier eignen sich spezielle 3D-Modelle, bei denen lediglich die Konstruktionspunkte durch Linien verbunden dargestellt sind (*Drahtgittermodelle*). 3D-Modelle werden aber nicht nur zur Nachbildung von Laborgeräten verwendet. Es bietet sich an, auch die einzelnen Arbeitsflächen und das virtuelle Labor selbst, das heißt die Laborräume und die Inneneinrichtung, mit Hilfe von 3D-Modellen nachzubilden und in das virtuelle Labor zu integrieren, wie zum Beispiel in *GenLab* (siehe Hasler und Schlattmann, 2001).

Des Weiteren werden die für das virtuelle Labor benötigten Audio- und Videosequenzen von den *Audio- und Videospezialisten*^{Rolle} erstellt (siehe dazu Kerres, 1998, Seite 334-338). Zur Erstellung von Audiosequenzen können die Geräusche der realen Versuchskomponenten aufgezeichnet werden. Diese dienen dann zur Vertonung der virtuellen Versuchskomponenten. Des Weiteren kann zum Beispiel für die Erstellung von Sprache ein professioneller Sprecher beauftragt werden (Merx, 1999, Seite 127). Soll die Darstellung eine besonders hohe Realitätstreue haben, so können außerdem Reaktionsabläufe in einem realen Labor mit einer Videokamera aufgezeichnet und als Videosequenz in das virtuelle Labor eingebunden werden. Bei der Erstellung von Videosequenzen für mehrsprachige virtuelle Labore ist darauf zu achten, dass die Audiosequenzen der verschiedenen Sprachversionen getrennt von der Videosequenz vorliegen, da Videosequenzen sehr speicherintensiv sind (Yass, 2000, Seite 262). Die Audiosequenz der entsprechenden Sprachversion wird dann gleichzeitig mit der Videosequenz abgespielt.

Des Weiteren werden in dieser Aktivität die einzelnen Graphiken des virtuellen Labors von den *Graphik-Designern*^{Rolle} erstellt. Dazu werden entsprechende Bildbearbeitungsprogramme eingesetzt.

7.5.8. Beschaffe die Medien

Der *Projektleiter*^{Rolle} wählt anhand der Anforderungen an die zu beschaffenden Medien ein geeignetes Unternehmen aus, dass die geforderte Leistung innerhalb eines bestimmten Zeit- und Kostenrahmens erbringen kann. Danach wird eine Vereinbarung über die erforderliche Qualität der Medien erstellt und in einem Vertrag schriftlich festgehalten. Außerdem werden die Lieferbedingungen mit dem Unternehmen ausgehandelt und die Erstellung der Medien in Auftrag gegeben.

7.5.9. Prüfe die Medien

Bevor die Medien für die Integration freigegeben werden, sind diese durch die *Medienspezialisten*[Rolle] zu prüfen. Dazu sind zum einen die korrekten technischen Daten, wie Auflösung, Farbtiefe und Format zu kontrollieren. Zum anderen sind die gestalterischen Aspekte zu berücksichtigen. Dazu ist das Erscheinungsbild des Mediums im Gesamtkontext des virtuellen Labors zu beurteilen. Zu betrachtende Aspekte sind beispielsweise die Farbwahl und die Realitätstreue der Modellierung. Ein voraussichtliches Erscheinungsbild im Gesamtkontext des virtuellen Labors ist dann zu beurteilen, wenn die Medien für ein Laborgerät zwar bereits erstellt, aber noch nicht integriert worden sind. Setzt sich ein Medium aus mehreren Medien zusammen, wie zum Beispiel bei den Abbildern der Laborgeräte in der 2D-Darstellung, so muss geprüft werden, ob sich die einzelnen Teile zu einem sinnvollen Ganzen zusammenfügen lassen. Dazu werden die einzelnen Medien in ein geeignetes Bildbearbeitungsprogramm[17] geladen, zusammengesetzt und insgesamt bewertet, das heißt es wird geprüft, ob die Medien zueinander passen. Medien die nicht in Ordnung sind, müssen erneut digitalisiert, überarbeitet oder erstellt werden.

Wurde ein externes Unternehmen mit der Erstellung der Medien beauftragt (siehe *Beschaffe die Medien*[Aktivität] in Kapitel 7.5.8), dann werden nach Lieferung der Medien diese von den *Medienspezialisten*[Rolle] geprüft. Entsprechen die Medien nicht der vereinbarten Leistung, so wird dies dem Hersteller mitgeteilt und eine Nachbesserung gefordert. Sind die Medien in Ordnung, so erfolgt die Abnahme durch den *Projektleiter*[Rolle]. Unter Umständen müssen die Medien nach Lieferung und Prüfung noch weiter verarbeitet werden. Das ist zum Beispiel dann notwendig, wenn kleine Detailänderungen an den Medien vorgenommen werden müssen oder wenn die Medien in einer sehr hohen Qualität geliefert werden, die für den Einsatz im virtuellen Labor nicht praktikabel ist. Das ist beispielsweise dann der Fall, wenn während der Integration der Medien technische Probleme auftreten, wie zum Beispiel dass nicht genügend Speicher für eine Videosequenz zur Verfügung steht.

7.6. Implementierung

Die Struktur des in Abbildung 7.23 dargestellten Workflows der *Implementierung*[Workflow] entspricht im Wesentlichen dem Workflow der Implementierung nach dem Rational Unified Process (siehe Kruchten, 2000, Seite 190). Neben der eigentlichen Implementierung und der Integration der Komponenten und Teilsysteme des virtuellen Labors, umfasst der Workflow im Gegensatz zum Rational Unified Process jedoch auch die Integration der in der *Medienproduktion*[Workflow] (siehe Kapitel 7.5) hergestellten Medien.

Zu Beginn des Workflows wird die Implementierung strukturiert, das heißt es wird festgelegt, in welcher Reihenfolge die Komponenten des virtuellen Labors implementiert werden. Darauf basierend wird die Integration der Komponenten und der Medien geplant. Danach werden die Komponenten implementiert und die zugehörenden Medien integriert.

[17] Zum Beispiel *Adobe Photoshop* von *Adobe* (siehe: http://www.adobe.com/products/photoshop/main.html)

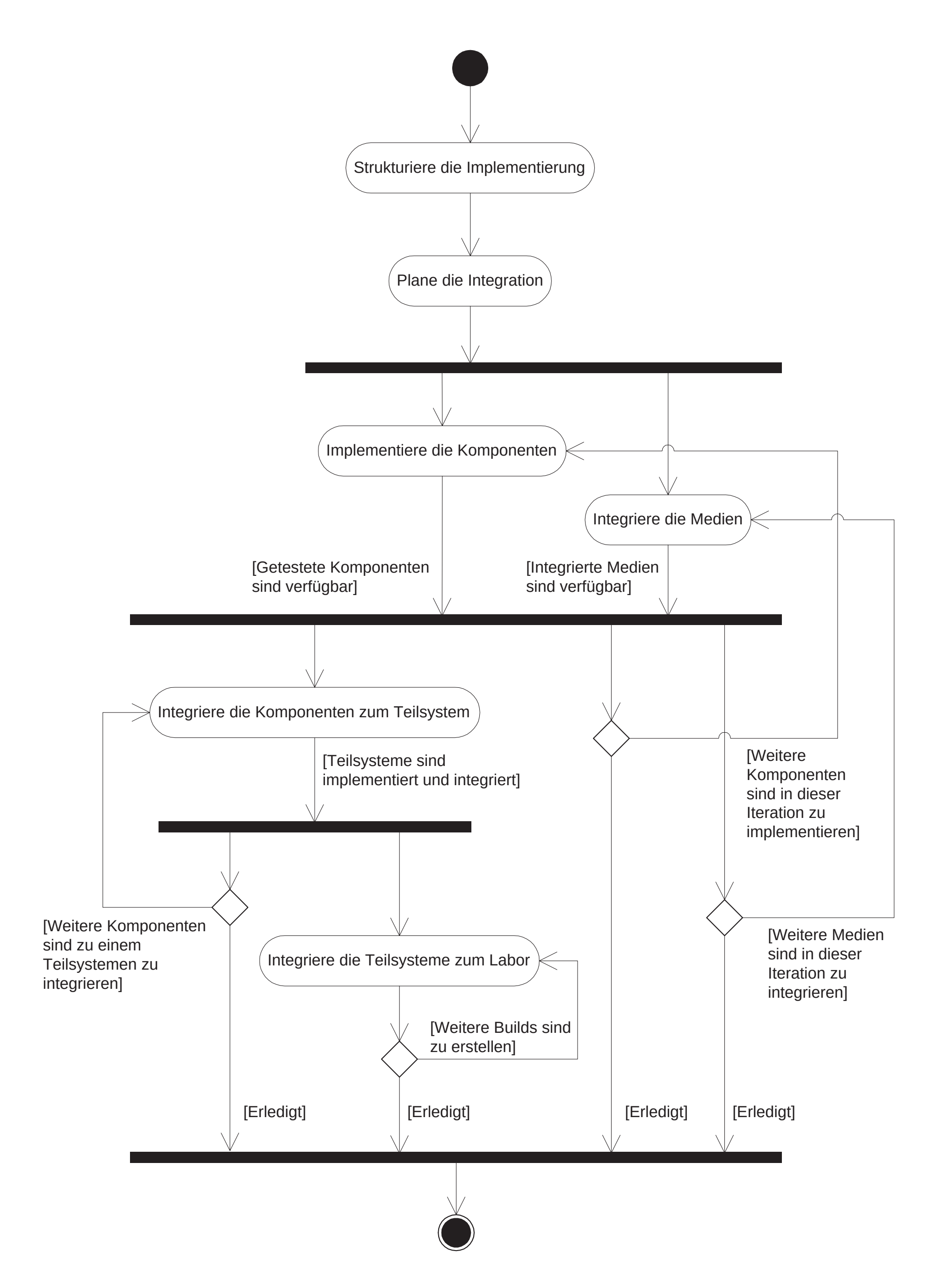


Abbildung 7.23.: Der Workflow für die Implementierung

Dieses wird solange fortgeführt, bis in der Iteration keine weiteren Komponenten mehr zu implementieren sowie keine weiteren Medien mehr zu integrieren sind. Parallel dazu werden die implementierten Komponenten zu Teilsystemen integriert. Anschließend werden die so entstandenen Teilsysteme zum virtuellen Labor integriert.

Die Aktivitäten der *Implementierung*^Workflow werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Strukturiere die Implementierung*[Aktivität] (siehe Kapitel 7.6.1),
- *Plane die Integration*[Aktivität] (siehe Kapitel 7.6.2),
- *Implementiere die Komponenten*[Aktivität] (siehe Kapitel 7.6.3),
- *Integriere die Medien*[Aktivität] (siehe Kapitel 7.6.4),
- *Integriere die Komponenten zum Teilsystem*[Aktivität] (siehe Kapitel 7.6.5) und
- *Integriere die Teilsysteme zum Labor*[Aktivität] (siehe Kapitel 7.6.6).

7.6.1. Strukturiere die Implementierung

Diese Aktivität führen die *Designer/Entwickler*[Rolle] zusammen mit den *Medienspezialisten*[Rolle] unter Koordination des *Projektleiters*[Rolle] durch. Auf der Basis des *OOD-Modells*[Artefakt] wird eine Strukturierung der Implementierung vorgenommen, so dass es zu minimalen Konflikten zwischen den zu implementierenden Klassen, den zu integrierenden Medien und den zu erstellenden *Builds*[18] kommt. Für die Implementierung wird das Entwicklerteam in parallel arbeitende Gruppen aufgeteilt und es wird festgelegt, wann welche Komponenten[19] des virtuellen Labors implementiert werden. Beispiele für Komponenten sind die Arbeitsflächen, Laborgeräte, Behälter und Substanzen, die Module des intelligenten Tutorsystems sowie das Anleitungsfenster und die Transportleiste (siehe *Entwerfe die Komponenten*[Aktivität] in Kapitel 7.4.4). Aber auch die Klassen zur Anbindung einer Datenbank oder eines Expertensystems stellen Komponenten dar. Zur Strukturierung der Implementierung gilt generell, dass zunächst die benötigten Arbeitsflächen zu implementieren sind. Danach werden die Versuchskomponenten, also die Laborgeräte, Behälter und Substanzen zu den Arbeitsflächen erstellt. Abschließend wird der eigentliche Versuchsablauf implementiert, das heißt das Versuchsprotokoll umgesetzt. Ergebnis der Strukturierung der Implementierung ist der *Implementierungsplan*[Artefakt] der anstehenden Iteration.

7.6.2. Plane die Integration

In diesem Teil der *Implementierung*[Workflow] erstellen die *Designer/Entwickler*[Rolle] und *Medienspezialisten*[Rolle] auf der Basis des *OOD-Modells*[Artefakt] und des *Implementierungsplans*[Artefakt] den *Integrationsplan*[Artefakt]. Der *Integrationsplan*[Artefakt] legt fest, wann welche Komponenten, Medien und Teilsysteme des virtuellen Labors integriert werden sollen. Des Weiteren sind gegebenenfalls die Inhalte für die Datenbank oder das Expertensystem zu integrieren beziehungsweise einzuspielen. Außerdem wird im *Integrationsplan*[Artefakt] festgelegt, wann ein *Build* des virtuellen Labors zu erstellen ist. Allgemein gilt, dass zumindest immer nach der Integration einer neuen Komponente in das virtuelle Labor ein neues *Build* zu erstellen ist.

[18] Siehe Glossar
[19] Auch: *Halbfabrikat* (siehe Balzert, 2000b, Seite 856)

7.6.3. Implementiere die Komponenten

Die *Designer/Entwickler*[Rolle] und *Simulationsspezialisten*[Rolle] implementieren die Komponenten des virtuellen Labors anhand des *Implementierungsplans*[Artefakt] der Iteration. Dabei sind die *Programmierrichtlinien*[Artefakt] des virtuellen Labors zu berücksichtigen und einzuhalten (siehe dazu *Konzipiere die Leitlinien für die Iteration*[Aktivität] in Kapitel 7.10.3). Generell gilt für die Implementierung einer Komponente, dass entsprechende *Model*-, *View*- und *Controller*-Klassen zu implementieren sind. Handelt es sich bei der Komponente um ein komplexes Simulationsmodell beziehungsweise komplexe Reaktionsabläufe, das höhere Anforderungen an die Implementierung stellt, so bietet sich die Praktik des Programmierens in Paaren an. Ziel dieser Praktik ist es, die Korrektheit des Simulationsmodells sicherzustellen. Werden die Versuchsabläufe nicht mit Hilfe eines Entwicklungswerkzeuges erstellt (siehe *Spezifiziere Aufbau und Ablauf der Lerneinheit*[Aktivität] in Kapitel 7.3.4), so wird der Versuch anhand des *annotierten Versuchsprotokolls*[Artefakt] implementiert und ein entsprechendes Anleitungsfenster dazu erstellt. Für die Implementierung von Mehrsprachigkeit sind außerdem entsprechende Klassen der Programmiersprache zu verwenden, wie zum Beispiel die Internationalisierung bei Java[20], oder es sind entsprechende Abfragen nach der ausgewählten Sprachversion in die Quelltexte des virtuellen Labors einzufügen (siehe dazu Yass, 2000, Seite 267).

Die implementierten Klassen und Komponenten werden anschließend von den *Designern/Entwicklern*[Rolle] und den *Simulationsspezialisten*[Rolle] getestet sowie Fehler behoben. Es werden alle Funktionalitäten der Laborgeräte sowie sämtliche Interaktionsmöglichkeiten zwischen den Laborgeräten, Behältern und Substanzen getestet. Dabei können die Tests der *Model*-Klassen bereits auf Kommandozeilenebene durchgeführt werden (siehe Boles u. a., 1998, Seite 46). Das Ergebnis dieser Aktivität sind also implementierte und getestete Komponenten des virtuellen Labors.

Das Framework des virtuellen Labors ist nach den Erfahrungen von *GenLab* so zu entwickeln, dass bei dem Entwurf und der Implementierung des Frameworks bereits sämtliche Funktionalitäten, das heißt sämtliche Steuerungs- und Konfigurationsmöglichkeiten berücksichtigt und integriert werden. Das bedeutet konkret, dass sämtliche Knöpfe und Schalter der virtuellen Laborgeräte und Behälter direkt angesprochen werden können und die Unterstützung für die Lerneranalyse und die Mehrsprachigkeit bereits integriert ist. Eine spätere Erweiterung des Frameworks um das Sperren einzelner Knöpfe und Schalter von Laborgeräten, die Erweiterung um eine Lerneranalyse oder die Unterstützung von Mehrsprachigkeit ist sehr aufwendig und daher sehr teuer.

7.6.4. Integriere die Medien

In diesem Teil der *Implementierung*[Workflow] werden die Medien von den *Designern/Entwicklern*[Rolle] oder den *Medienspezialisten*[Rolle] in die Komponenten des virtuellen Labors integriert. Dazu werden die einzelnen Medien unterschiedlichen Medientyps zueinander in Beziehung gesetzt (Steinmetz, 1999, Seite 838). So entsteht zum Beispiel aus einer

[20] Siehe: http://java.sun.com/j2se/1.3/docs/guide/intl/index.html

implementierten Komponente und einer Reihe von Graphiken und Audiosequenzen ein funktionsfähiges Abbild eines virtuellen Laborgerätes (siehe Boles u. a., 1998). Die *Medienspezialisten*[Rolle] stehen außerdem für technische Fragen bezüglich der Integration der Medien zur Verfügung.

Ist aus mehreren Medien ein Abbild eines virtuellen Laborgerätes oder Behälters zu erstellen, so bietet sich eine automatische Integration der Medien mit Hilfe eines entsprechenden Werkzeuges an. Dabei übernimmt das Werkzeug die technische Integration der Medien in das virtuelle Labor. Es ist dabei zu prüfen, ob sich die Medien zu einem korrekten Abbild des Laborgerätes oder Behälters zusammensetzen lassen und ein sinnvolles Gesamtbild ergeben. Außerdem kann mit dem Werkzeug jedem Zustand des Laborgerätes oder Behälters eine entsprechende Visualisierung zugeordnet werden (vergleiche das Werkzeug *GUI Composer* in Aden u. a., 1999, Seite 41).

Können die Medien aus irgendeinem technischen Grund nicht integriert werden oder sind diese nicht rechtzeitig verfügbar, so werden anstelle derer zunächst *Wegwerf-Medien* (siehe *Ermittle benötigte Medien*[Aktivität] in Kapitel 7.5.2) erstellt und integriert. Diese werden dann später durch die eigentlichen Medien ersetzt.

7.6.5. Integriere die Komponenten zum Teilsystem

Die *Designer/Entwickler*[Rolle] und *Simulationsspezialisten*[Rolle] integrieren die implementierten und getesteten Komponenten des virtuellen Labores nach dem *Integrationsplan*[Artefakt] der Iteration zu einem Teilsystem. Das bedeutet konkret, dass die neuen Arbeitsflächen, Laborgeräte und Behälter entsprechend ihrer *Model*-, *View*- und *Controller*-Klassen in das Framework, genauer gesagt in den Simulator integriert werden. Des Weiteren wird der Simulator um neue Reaktionsabläufe erweitert. Auch werden das Anleitungsfenster und die Transportleiste in das Framework integriert. Außerdem wird aus dem Lehrer-, Lerner- und Expertenmodul das intelligente Tutorsystem zusammengesetzt. Ergebnis dieser Aktivität sind die beiden implementierten Teilsysteme Simulator und intelligentes Tutorsystem, die getestet werden können.

7.6.6. Integriere die Teilsysteme zum Labor

Die *Designer/Entwickler*[Rolle] integrieren nach den Vorgaben des *Integrationsplans*[Artefakt] der Iteration die zuvor implementierten Teilsysteme zum virtuellen Labor. Ergebnis dieser Aktivität ist ein neues *Build* des virtuellen Labors, das getestet werden kann. Beispiele für Teilsysteme des virtuellen Labors sind der Simulator und das intelligente Tutorsystem. Aber auch ein an das virtuelle Labor angebundenes klassisches Lehr- und Lernsystem stellt ein Teilsystem dar.

7.7. Test und Evaluation

Der in der Abbildung 7.24 dargestellte Workflow *Test und Evaluation*[Workflow] orientiert sich nach dem Workflow Test des Rational Unified Process (siehe Kruchten, 2000, Seite 203) und wurde um die Evaluation virtueller Labore erweitert. Das Testen umfasst sämtliche funktionalen Tests, mit dem einzigen Zweck, Fehler zu entdecken (Appelrath und Ludewig, 2000; Frühauf u. a., 1991a). Die Evaluation des virtuellen Labors umfasst die Prüfung der gestalterischen Qualität der Benutzungsoberfläche (Sawhney, 1995, Seite 50), die didaktische Qualität des intelligenten Tutorsystems und die fachliche Korrektheit der Inhalte.

Zunächst werden im Workflow *Test und Evaluation*[Workflow] die in der aktuellen Iteration anstehenden Tests und die Evaluation des virtuellen Labors geplant. Anschließend werden die Tests und die Evaluation entworfen und implementiert. Danach teilt sich der Workflow *Test und Evaluation*[Workflow] in zwei parallele Pfade. Im linken Pfad werden die funktionalen Tests durchgeführt. Dazu werden zunächst die Teilsysteme des virtuellen Labors und anschließend das virtuelle Labor selbst getestet. Außerdem wird parallel dazu für jeden durchgeführten Test eine Bewertung der Testergebnisse vorgenommen. Im rechten Pfad erfolgt die Evaluation des virtuellen Labors. Danach werden die Ergebnisse der Evaluation analysiert. Nach Abschluss aller Testdurchläufe und der Analyse der Evaluation wird außerdem eine Zusammenfassung der Testbewertungen und der Evaluationsergebnisse erstellt, sowie eventuelle Änderungen an den Anforderungen des virtuellen Labors identifiziert.

Die Aktivitäten des Workflow *Test und Evaluation*[Workflow] werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Plane die Tests und Evaluation*[Aktivität] (siehe Kapitel 7.7.1),
- *Entwerfe die Tests und Evaluation*[Aktivität] (siehe Kapitel 7.7.2),
- *Implementiere die Tests und Evaluation*[Aktivität] (siehe Kapitel 7.7.3),
- *Teste die Teilsysteme des Labors*[Aktivität] (siehe Kapitel 7.7.4),
- *Teste das Labor*[Aktivität] (siehe Kapitel 7.7.5),
- *Auswerten des Tests*[Aktivität] (siehe Kapitel 7.7.6),
- *Evaluiere das virtuelle Labor*[Aktivität] (siehe Kapitel 7.7.7),
- *Analysiere Ergebnisse der Evaluation*[Aktivität] (siehe Kapitel 7.7.8) und
- *Erstelle Zusammenfassung der Tests und Evaluation*[Aktivität] (siehe Kapitel 7.7.9).

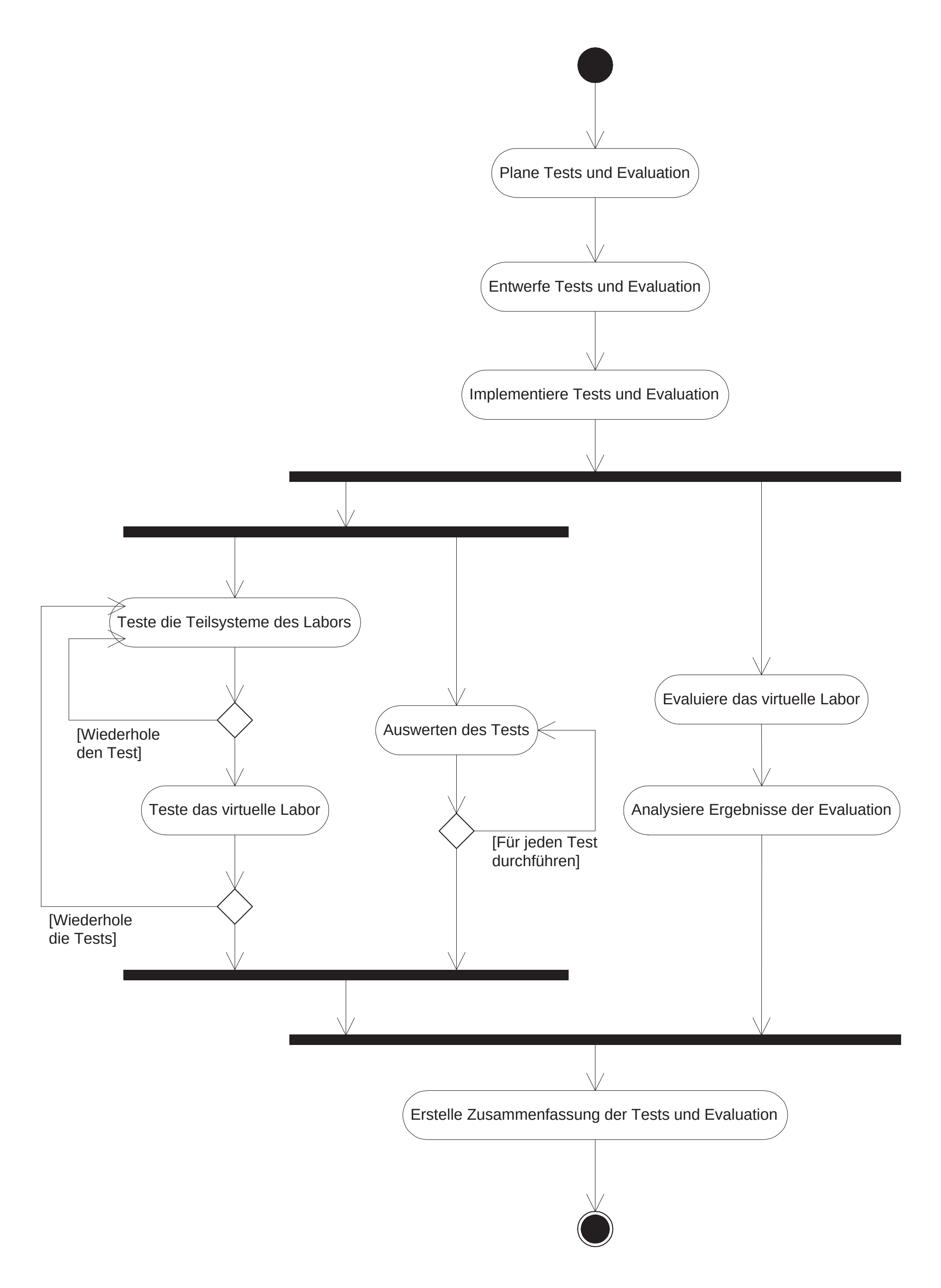


Abbildung 7.24.: Der Workflow zu Test und Evaluation

7.7.1. Plane die Tests und Evaluation

In dieser Aktivität werden die Tests und die Evaluation des virtuellen Labors geplant (siehe Abbildung 7.25). Dazu erstellen die *Tester*^{Rolle} einen *Test- und Evaluationsplan*^{Artefakt} für das virtuelle Labor. Bei der Planung der Tests geht es um die Festlegung einer entsprechenden Teststrategie. Diese bestimmt, in welcher Reihenfolge die Tests ausgeführt

werden (Frühauf u. a., 1991b, Seite 83). Dafür werden die *Testfälle* zum virtuellen Labor erhoben, die in der aktuellen Iteration getestet werden sollen. Jeder Testfall wird mittels einer *Testprozedur* beschrieben. Zu jeder Testprozedur gehört außerdem eine *Testkonfiguration*, die die Rahmenbedingungen für den Testfall beschreibt. Bei der Planung der Evaluation geht es um die Festlegung der Mittel zur Evaluation der gestalterischen, didaktischen und fachlichen Qualität des virtuellen Labors. Dabei sind insbesondere die Lernerakzeptanz und der Lernerfolg zu bestimmen. Die Planung der Tests und der Evaluation endet mit der Bestätigung des *Test- und Evaluationsplans*^Artefakt durch den *Projektleiter*^Rolle.

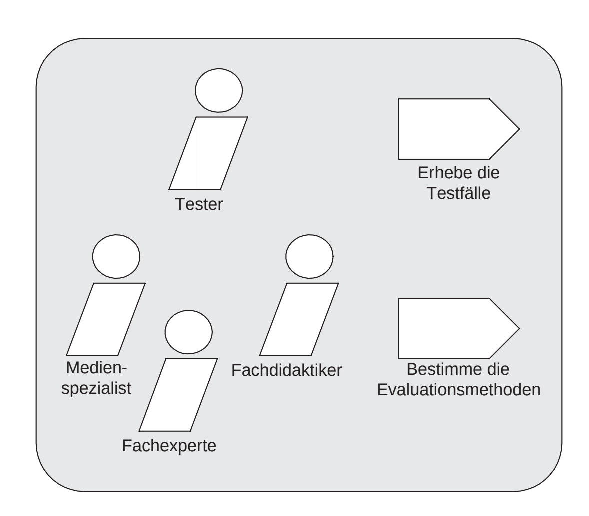


Abbildung 7.25.: Detailansicht der Aktivität Plane die Tests und Evaluation

7.7.1.1. Erhebe die Testfälle

Nach (Frühauf u. a., 1991b, Seite 42) stehen für die Erhebung der Testfälle prinzipiell drei Informationsquellen zur Verfügung. Das sind die Anwendungsfälle, die Quelltexte beziehungsweise das *OOD-Modell*^Artefakt des virtuellen Labors sowie die Erfahrungswerte des Entwicklerteams über die häufigsten Software-Fehler. Die aus den Anwendungsfällen abgeleiteten Testfälle müssen dabei folgende Anforderungen erfüllen:

- Jede Funktion des virtuellen Labors wird mindestens einmal aktiviert. Das heißt alle Versuche und Fertigkeiten werden durchlaufen, das intelligente Tutorsystem wird aufgerufen und die Navigation im virtuellen Labor durchgeführt. Es werden alle möglichen Alternativen bei den Versuchsabläufen getestet, alle denkbaren Interaktionen zwischen den Laborgeräten, Behältern und Substanzen durchgeführt und sämtliche Reaktionsabläufe des Simulators geprüft.

- Alle möglichen Ausgabeformen des virtuellen Labors werden mindestens einmal erzeugt. Die hauptsächliche Ausgabeform ist die Darstellung des virtuellen Labors auf dem Bildschirm. Weitere Ausgabeform ist der Ausdruck auf Papier, zum Beispiel von Versuchsprotokollen.

- Die Leistungsgrenzen des virtuellen Labors sind auszuloten. So ist zum Beispiel die Reaktionszeit des virtuellen Labors auf die Eingaben des Lerners zu untersuchen. Dazu sind die Ressourcen des Computer-Systems, auf dem das virtuelle Labor getestet wird, zu variieren.

Werden die Quelltexte beziehungsweise das *OOD-Modell*[Artefakt] des virtuellen Labors zur Erhebung der Testfälle betrachtet, so muss durch die Testfälle sichergestellt werden, dass jeder (Programm-)Zweig der einzelnen Klassen mindestens einmal durchlaufen wird (*Zweigüberdeckungstest* beziehungsweise C_1-*Test*). Der Zweigüberdeckungstest gilt als minimales Testkriterium (siehe Balzert u. a., 1993; Liggesmeyer, 1993, 1990) und schließt den *Anweisungsüberdeckungstest* beziehungsweise C_0-*Test* mit ein (Balzert u. a., 1993, Seite 35). In diesem wird nur die mindestens einmalige Ausführung aller Anweisungen der Klassen des Software-Systems gefordert (Balzert u. a., 1993, Seite 31 f.). Ein etwas stärkeres Kriterium als der Zweigüberdeckungstest ist der *Bedingungsüberdeckungstest*, bei dem gefordert wird, dass alle Teilausdrücke der logischen Bedingungen in den Programmverzweigungen mindestens einmal berücksichtigt werden (Appelrath und Ludewig, 2000, Seite 233). Da virtuelle Labore besonders interaktive Software-Systeme darstellen, ist die Forderung des *Pfadüberdeckungstest* (siehe Liggesmeyer, 1993, Seite 337), dass alle unterschiedlichen vollständigen Pfade des virtuellen Labors durchlaufen werden, wegen der hohen Anzahl verschiedener Pfade unrealistisch (siehe dazu Appelrath und Ludewig, 2000, Seite 233). Ein ablauffähiger Quelltext alleine garantiert jedoch noch nicht, dass auch alle Anforderungen an das virtuelle Labor erfüllt werden. Daher werden insbesondere auch die Schnittstellen zur Erhebung von Testfällen herangezogen, da sich nur so fehlende beziehungsweise unberücksichtigte Anforderungen entdecken lassen.

Die dritte Quelle zur Erhebung von Testfällen bezieht sich auf die Erfahrungswerte des Entwicklerteams über die häufigsten Software-Fehler. Das sind zum Beispiel provozieren von eventuell nicht berücksichtigten Fehlerzuständen. Dazu gehört auch das Weglassen bestimmter Gerätetreiber, die vom virtuellen Labor benötigt werden, oder die Verwendung älterer Versionen. Weitere Fehlerzustände werden erzeugt, wenn das Kabel zum Drucker herausgezogen, die CD-ROM oder Diskette aus dem Laufwerk entnommen oder wenn die Netzwerkverbindung getrennt wird.

Wichtig bei der Identifikation der Testfälle ist, dass alle Anwendungsfälle des *Anwendungsfall-Modells*[Artefakt] abgedeckt werden. Da ein virtuelles Labor ein hochgradig interaktives multimediales Lehr- und Lernsystem darstellt, werden hauptsächlich dynamische Tests durchgeführt, das heißt das virtuelle Labor wird auf einem Computer-System ausgeführt und getestet (siehe dazu Balzert u. a., 1993; Liggesmeyer, 1993, 1990). Statische Analysen dagegen eignen sich bei virtuellen Laboren zum Beispiel zur Überprüfung des Simulators. Dabei wird die Korrektheit des Simulators anhand eines Zustandsübergangsgraphen oder Zustandsdiagramms des Simulationsmodells geprüft. Des Weiteren lassen sich statische Analysen auch für die einzelnen Laborgeräte anhand entsprechender Zustandsdiagramme durchführen (siehe Boles u. a., 1998, Seite 45). Aber auch die dynamischen Tests des virtuellen Labors lassen sich mit Hilfe der Zustandsdiagramme planen. Nach (Appelrath und Ludewig, 2000, Seite 225) sollten die Tests nicht von den *Projektmitarbeitern*[Rolle] durchgeführt werden,

die direkt an der Entwicklung der entsprechenden Programmteile beteiligt waren, da diese sich meist unbewusst dagegen sträuben, Fehler zu finden. Erfolgt die Entwicklung in mehreren Gruppen, so testen die *Projektmitarbeiter*[Rolle] die entsprechenden Programmteile gegenseitig.

7.7.1.2. Bestimme die Evaluationsmethoden

Bezüglich der Evaluation des virtuellen Labors sind die gestalterischen, didaktischen und fachlichen Aspekte zu bewerten. Diese spiegeln sich in den folgenden Evaluationskriterien und -zielen wider:

- Jede Lerneinheit, das heißt jeder Versuch und jede Fertigkeit wird hinsichtlich der didaktischen Aspekte geprüft (didaktisches Erproben). Bezüglich des *didaktischen Konzeptes*[Artefakt] ist zu prüfen, ob die Lerneinheiten inhaltliche Lücken aufweisen und ob die Formulierungen und Darstellungen des intelligenten Tutorsystems klar und eindeutig sind (Förster und Zwernemann, 1993, Seite 73 f.). Des Weiteren ist die Lernerakzeptanz und der Lerneffekt des virtuellen Labors zu ermitteln.

- Jede Lerneinheit, die entsprechende Hintergrundinformation und die Inhalte des intelligenten Tutorsystems werden auf fachliche Korrektheit hin geprüft. Insbesondere wird geprüft, ob sich der Simulator so verhält, wie in der Realität, das heißt ob das Simulationsmodell korrekt ist.

- Die gestalterische Qualität der Benutzungsoberfläche und der Dialogführung ist zu bewerten (vergleiche Sawhney, 1995, Seite 50). Dazu werden die Medien bezüglich Qualität und Gestaltung bewertet und es wird das Zusammenspiel der Medien im zeitlichen Verlauf betrachtet. Wurden vom *Auftraggeber*[Rolle] Vorgaben des *Corporate Design* definiert, so wird deren Einhaltung geprüft.

Zur Beurteilung der Evaluationskritieren und -ziele stehen eine Reihe von Evaluationsmethoden zur Verfügung. Diese lassen sich nach dem Grad dessen, inwieweit bei der Beurteilung der Evaluationskriterien die *Endanwender*[Rolle] beteiligt sind, unterscheiden (siehe Oppermann und Reiterer, 1994, Seite 342 bis 345):[21]

- *Subjektive Evaluationsmethoden*: Bei den subjektiven Evaluationsmethoden werden die Evaluationskriterien von den *Endanwendern*[Rolle] beurteilt. Diese werden weiter unterschieden in:
  - *Mündliche Befragung*: Es werden dem *Endanwender*[Rolle] Fragen gestellt, die zur Beurteilung der gewählten Evaluationskriterien dienlich sind.
  - *Schriftliche Befragung*: Es werden ein oder mehrere (elektronische) Fragebögen erstellt und vom *Endanwender*[Rolle] ausgefüllt.

[21] Vergleiche dazu (Freibichler, 2000)

- *Lautem Denken*: Der *Endanwender*[Rolle] wird aufgefordert, während der Versuchsdurchführung seine Überlegungen, Probleme und Handlungsalternativen laut vorzutragen.

- *Objektive Evaluationsmethoden*: Im Gegensatz zu den subjektiven Evaluationsmethoden wird hier versucht, alle subjektiven Einflüsse der *Endanwender*[Rolle] wie zum Beispiel Emotionen, Vorlieben und Vorurteile weitgehend auszuschließen. Die objektiven Evaluationsmethoden werden weiter unterschieden in:
  - *Anwesende Beobachtung*: Der *Fachdidaktiker*[Rolle] sitzt direkt neben dem *Endanwender*[Rolle] und versucht anhand der beobachteten Handlungen, Fehler, Ausführungszeiten und anderer wahrgenommener Attribute die Evaluationskriterien entsprechend zu beurteilen.
  - *Abwesende Beobachtung*: Der *Endanwender*[Rolle] wird indirekt beobachtet, wie zum Beispiel durch eine Videoaufzeichnung oder die Protokollierung der Lernerdaten.

- *Leitfadenorientierte Evaluationsmethode*: Das virtuelle Labor wird möglichst objektiv entlang eines Prüfleitfadens beurteilt (siehe zum Beispiel Lottmann u. a., 2000). Ein *Endanwender*[Rolle] wird für die leitfadenorientierte Evaluationsmethode nicht benötigt.

Welche Evaluationsmethoden ausgewählt werden ist von den Evaluationskriterien und -zielen abhängig. Für die Ermittlung der Lernerakzeptanz können zum Beispiel Fragebögen eingesetzt werden. Fragebögen können aber auch als *Lernzielkontrollen* zur Bestimmung des Lerneffekts dienen. Ebenfalls zur Bestimmung des Lerneffekts kann die Protokollierung der Lernerdaten eingesetzt werden. Zur Evaluation der gestalterischen Aspekte und der Vorgaben des *Corporate Designs* eignet sich zum Beispiel die leitfadenorientierte Evaluationsmethode.

7.7.2. Entwerfe die Tests und Evaluation

Unter Zuhilfenahme des *OOD-Modells*[Artefakt] und der implementierten Komponenten entwerfen die *Designer/Entwickler*[Rolle] die Testprozeduren zu den Testfällen des virtuellen Labors. Die Testprozeduren beschreiben zu den einzelnen Testfällen, wie diese durchgeführt werden. Beispiele für Testprozeduren sind die automatisierten Tests des Simulators. Die Ergebnisse der Tests werden dabei aufgezeichnet und anschließend von den *Testern*[Rolle] ausgewertet (vergleiche Sawhney, 1995, Seite 50). Weitere Beispiele sind Testprozeduren zur Überprüfung der Anbindung des virtuellen Labors an eine Datenbank oder die Auswertung der protokollierten Lernerdaten des intelligenten Tutorsystems. Zu jeder Testprozedur wird außerdem eine Testkonfiguration festgelegt. Eine Testkonfiguration beschreibt die exakten Einstellungen des virtuellen Labors, die vor dem Test einzunehmen sind. Im Falle eines automatisierten Tests ist die Testkonfiguration Bestandteil der zugehörigen Testprozedur.

Für die Evaluation des virtuellen Labors, also zur Prüfung der gestalterischen, didaktischen und fachlichen Aspekte, werden entsprechende Testprozeduren in nicht-formaler textueller

oder graphischer Form entworfen. Die *Medienspezialisten*[Rolle] entwerfen zum Beispiel Kriterienkataloge zur Beurteilung der gestalterischen Aspekte der Benutzungsoberfläche. Von den *Fachdidaktikern*[Rolle] werden Leitfäden zur Analyse der Benutzungsoberfläche und der Lerneinheiten aus didaktischer Sicht entworfen. Außerdem entwerfen die *Fachdidaktiker*[Rolle] Fragebögen, die als Vor- beziehungsweise Nachtest zur Evaluation des Lerneffektes dienen (siehe *Evaluiere das virtuelle Labor*[Aktivität] in Kapitel 7.7.7). Die *Fachexperten*[Rolle] erstellen Leitfäden zur Bestimmung der fachlichen Korrektheit der Versuche, Fertigkeiten und entsprechenden Hintergrundinformationen des virtuellen Labors.

7.7.3. Implementiere die Tests und Evaluation

Die *Designer/Entwickler*[Rolle] implementieren die Testprozeduren für der funktionalen Tests, soweit dies möglich ist. Ein Beispiel für Testprozeduren, die in dieser Aktivität implementiert werden, sind die automatisierten Tests des Simulators. Diese können für die Kommandozeilenebene erstellt und dort ausgeführt werden.

Für die Evaluation des virtuellen Labors werden zum Beispiel elektronische Fragebögen in Form eines klassischen Lehr- und Lernsystems erstellt. Dabei können entsprechende Entwicklungswerkzeuge eingesetzt werden, wie zum Beispiel *WebXam* der Universität Bielefeld (siehe http://www.webxam.de).

7.7.4. Teste die Teilsysteme des Labors

In dieser Aktivität werden die einzelnen Teilsysteme des virtuellen Labors, das heißt das Zusammenspiel der Komponenten der Teilsysteme getestet. Dazu führen die *Tester*[Rolle] gemäß dem *Test- und Evaluationsplan*[Artefakt] die Testprozeduren für die einzelnen Teilsysteme des virtuellen Labors durch. Getestet wird anhand des aktuellen *Build* des virtuellen Labors. Auftretende Fehler werden im *Test- und Evaluationsbericht*[Artefakt] entsprechend vermerkt.

7.7.5. Teste das Labor

Diese Aktivität wird analog zu der Aktivität *Teste die Teilsysteme des Labors*[Aktivität] in Kapitel 7.7.4 durchgeführt. Dabei betreffen die Testfälle jedoch nicht die Teilsysteme, sondern das virtuelle Labor selbst. Es werden die Interaktionen zwischen den einzelnen Teilsystemen des virtuellen Labors getestet, das heißt ob der Simulator und das intelligente Tutorsystem problemlos miteinander agieren, oder der Abruf von Hintergrundwissen zu den Lerneinheiten und Versuchskomponenten funktioniert. Auch ist die Anbindung weiterer Teilsysteme an das virtuelle Labor, wie zum Beispiel ein klassisches Lehr- und Lernsystem, zu testen.

7.7.6. Auswerten des Tests

Die *Tester*[Rolle] analysieren und bewerten die Testergebnisse aus der Testprozedur zu dem Testfall und halten die daraus gewonnenen Erkenntnisse in dem *Test- und Evaluationsbericht*[Artefakt] fest. Diese Aktivität wird für jeden Testfall durchgeführt.

7.7.7. Evaluiere das virtuelle Labor

Es werden die in der Aktivität *Bestimme die Evaluationsmethoden*[Aktivität] (siehe Abschnitt 7.7.1.2) festgelegten Methoden zur Evaluation des virtuellen Labors angewendet. Das heißt es werden zum Beispiel die gestalterischen Aspekte mit Hilfe eines entsprechenden Leitfadens von den *Medienspezialisten*[Rolle] bewertet. Insbesondere werden die Vorgaben des *Corporate Designs* von dem *Auftraggeber*[Rolle] geprüft.

Zur Bestimmung des *Lerneffekts* werden *Vor- und Nachtests* durchgeführt. Dabei wird zunächst das Vorwissen einer Prüfgruppe mit Hilfe eines Vortests erfasst. Die Prüfgruppe setzt sich zusammen aus einer repräsentativen Anzahl an *Endanwendern*[Rolle] aus der Zielgruppe des virtuellen Labors. Der Vortest kann zum Beispiel ein elektronischer Fragebogen sein. Mit diesem Fragebogen können sowohl Fakten, als auch Handlungsabläufe geprüft werden. Nach dem Vortest wird das virtuelle Labor eingesetzt. Nach einer bestimmten Zeit endet die Lernphase und es wird mit einem zum Vortest parallelen Nachtest der Wissenstand der Prüfgruppe erneut ermittelt. Aus der Differenz zwischen Vor- und Nachtest wird der Lernerfolg der Lerner und damit die Lerneffektivität des virtuellen Labors bestimmt. Auf die Verwendung einer Vergleichsgruppe (*peergroup*) zu den Prüflingen wird in (Baumgartner, 1997, Seite 243) abgeraten, da selbst bei gegebener Vergleichbarkeit der beiden Gruppen durch die unterschiedliche Lernsituation (zum Beispiel reales Laborpraktikum verglichen mit virtuelles Labor) die Ergebnisse leicht verfälscht werden können. Um diese Zufälligkeiten auszugleichen, ist der Vergleich beider Lehr- und Lernformen mit einer entsprechend großen Stichprobe erforderlich (siehe auch Eiwan, 1999, Seite 81 ff.). Auch müssen andere Einflussfaktoren, wie zum Beispiel der *Hawthorne-Effekt* mit berücksichtigt werden. Beim Hawthorne-Effekt hat nach (Baumgartner, 1997, Seite 243) das multimediale Lehr- und Lernsystem nicht deshalb einen höheren Lerneffekt, weil es didaktisch besser gestaltet wurde als zum Beispiel ein Buch oder eine klassische Lehrveranstaltung, sondern weil es als neues Medium interessanter ist (siehe auch Preim, 1999, Seite 196 f.).

Außerdem ist bei der Evaluation des virtuellen Labors der *Lerntransfer* zu bewerten, das heißt es wird evaluiert, wie der Lerner die im virtuellen Labor erlernten Handlungsabläufe auf die Realität überträgt (siehe auch Psaralidis und Zimmer, 2000, Seite 272 ff.). Bei dem Lerntransfer geht es also nicht um die reine Behaltensleistung von Fakten, sondern um den Erwerb und die Anwendung von strategischem Wissen und Handlungskompetenz (vergleiche Kerres, 1998, Seite 116). Dazu wird mit Hilfe der anwesenden oder abwesenden Beobachtung evaluiert, wie sich ein Lerner, der sich zuvor mit Hilfe des virtuellen Labors auf das Laborpraktikum vorbereitet hat, anschließend bei der Durchführung der gleichen Versuche in der realen Laborumgebung zurechtfindet und verhält. Die Evaluation des

Lerntransfers wurde nach (Freibichler, 2000, Seite 307) bisher nur sehr selten durchgeführt, weil dazu insbesondere eine aufwendige und differenzierte Aufgaben-, Tätigkeits- und Problemanalyse vorausgesetzt wird. Sowohl eine differenzierte Beschreibung der Aufgaben des Laborpraktikanten (Versuche durchführen) und der jeweiligen Tätigkeiten (Arbeitsschritte des entsprechenden Versuchsprotokolls) sind für den Anwendungsfall virtuelle Labore verfügbar. Auch eine detaillierte Problemanalyse ist bei virtuellen Laboren mit der Erstellung der *annotierten Versuchsprotokolle*[Artefakt] und der Aktivitäts- und Zustandsdiagramme zu den Versuchen vorhanden. Damit steht der Analyse des Lerntransfers nur noch der relativ große Aufwand zur Durchführung entgegen, der jedoch im Verhältnis zum erwartbaren Nutzen als relativ gering erscheint.

7.7.8. Analysiere Ergebnisse der Evaluation

In dieser Aktivität werden die Ergebnisse aus den Befragungen und Beobachtungen der *Endanwender*[Rolle] durch die *Fachdidaktiker*[Rolle] ausgewertet. Des Weiteren werden die protokollierten Lernerdaten mit Hilfe von *Nachanalysen* aufbereitet (siehe dazu *Konzipiere die Lerneranalyse*[Aktivität] in Abschnitt 7.3.3.4). In den Nachanalysen ist nicht die Bestimmung des Lerneffekts und der damit verbundene Lernzuwachs von Interesse, sondern es wird der Lernprozess der Lerner analysiert (vergleiche Schulmeister, 1997, Seite 383). Die Lernerdaten lassen sich dazu zum Beispiel als *Verhaltensspur* darstellen, bei der die einzelnen Aktionen des Lerners in tabellarischer Form aufgelistet werden. Es ist aber ebenso denkbar, die protokollierten Lernerdaten als Animation (*Wiederholung*) im virtuellen Labor abspielen zu lassen. Können in der Nachanalyse der Lernerdaten explizite Wissenszustände des Lerners extrahiert werden, so ist es auch möglich, die Lernerdaten in Form eines textuellen Versuchsprotokolls, wie es von den *Fachexperten*[Rolle] als Vorlage zur Versuchsdurchführung erstellt wird, aufzubereiten.

7.7.9. Erstelle Zusammenfassung der Tests und Evaluation

In dieser Aktivität wird von den *Testern*[Rolle] eine Zusammenfassung aller Testbewertungen und der Ergebnisse der Evaluation erstellt und im *Test- und Evaluationsbericht*[Artefakt] festgehalten (vergleiche Frühauf u. a., 1991b, Seite 80). Außerdem werden eventuelle Änderungswünsche an das virtuelle Labor identifiziert. Anschließend werden die Änderungswünsche klassifiziert und an den *Projektleiter*[Rolle] weitergegeben.

7.8. Einsatz

In der Abbildung 7.26 ist der Workflow zum *Einsatz*[Workflow] des virtuellen Labors dargestellt. Gegenüber dem entsprechenden Workflow des Rational Unified Process konnte allerdings die Struktur vereinfacht werden (siehe Kruchten, 2000, Seite 233).

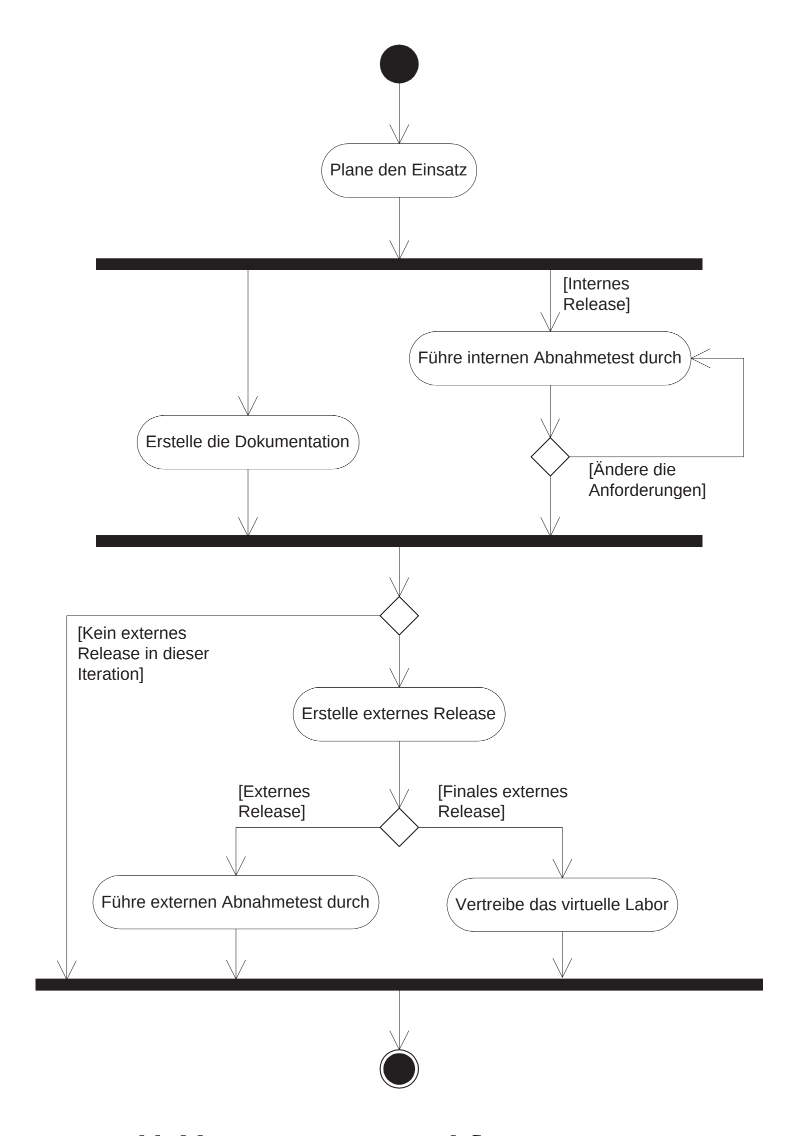


Abbildung 7.26.: Der Workflow zum Einsatz

Zunächst wird in dem Workflow der Einsatz des virtuellen Labors geplant. Danach wird die Dokumentation erstellt, sowie ein interner Abnahmetest durchgeführt. Der interne Abnahmetest wird solange wiederholt, bis alle Anforderungen an das virtuelle Labor stabil sind, das heißt bis sich keine Änderungen an den Anforderungen der aktuellen Iteration mehr ergeben. Anschließend wird, sofern im *Entwicklungsplan der Iteration*^Artefakt vorgesehen, ein *externes Release*^Artefakt erstellt und ein externer Abnahmetest durchgeführt. Handelt es sich um das finale *externe Release*^Artefakt, so beginnt der Vertrieb des virtuellen Labors.

Die Aktivitäten im Workflow *Einsatz*^Workflow werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Plane den Einsatz*^Aktivität (siehe Kapitel 7.8.1),
- *Erstelle die Dokumentation*^Aktivität (siehe Kapitel 7.8.2),

- *Führe internen Abnahmetest durch*[Aktivität] (siehe Kapitel 7.8.3),
- *Erstelle externes Release*[Aktivität] (siehe Kapitel 7.8.4),
- *Führe externen Abnahmetest durch*[Aktivität] (siehe Kapitel 7.8.5) und
- *Vertreibe das virtuelle Labor*[Aktivität] (siehe Kapitel 7.8.6).

7.8.1. Plane den Einsatz

Der *Projektleiter*[Rolle] plant auf der Basis des *Gesamtentwicklungsplans*[Artefakt] und des *Entwicklungsplans der Iteration*[Artefakt] wann und wie das nächste *Release* des virtuellen Labors erfolgen soll. Außerdem wird eine Liste der Materialien erstellt, die für das *Release* benötigt werden, wie zum Beispiel das *Booklet* zur CD-ROM und die Verpackung. Dabei fließen die Erfahrungen aus den vorherigen *Releases* in die Planung des nächsten *Releases* mit ein.

7.8.2. Erstelle die Dokumentation

Unter der *Dokumentation* wird die Gesamtheit der Unterlagen verstanden, die dem *Endanwender*[Rolle] einen ordnungsgemäßen Betrieb (Balzert, 2000b, Seite 1089) und dem *Auftraggeber*[Rolle] die Wartung und Weiterentwicklung des virtuellen Labors ermöglichen. Die Dokumentation zum virtuellen Labor besteht daher aus einem *Benutzungshandbuch*[Artefakt] für die *Endanwender*[Rolle] und einer *technischen Dokumentation*[Artefakt] für den *Auftraggeber*[Rolle] (siehe Abbildung 7.27).

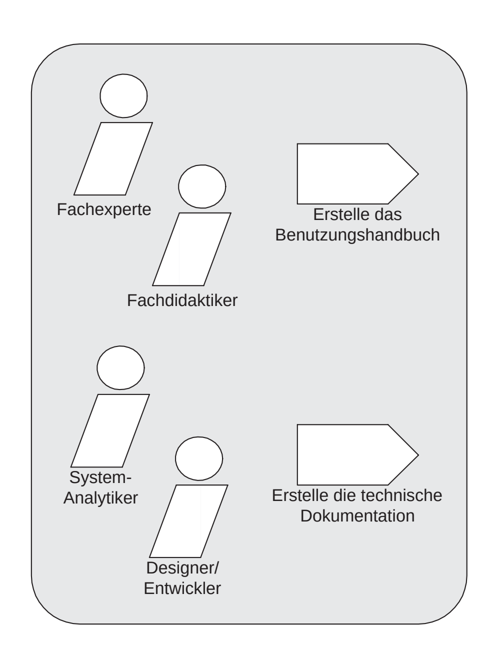


Abbildung 7.27.: Detailansicht der Aktivität Erstelle die Dokumentation

7.8.2.1. Erstelle das Benutzungshandbuch

Das *Benutzungshandbuch*[Artefakt] des virtuellen Labors wird gemeinsam von den *Fachdidaktikern*[Rolle] und den *Fachexperten*[Rolle] geschrieben. Prinzipiell kann zwischen einem gedruckten *Benutzungshandbuch*[Artefakt] und einem elektronischen *Benutzungshandbuch*[Artefakt] (*Online-Handbuch* oder auch *multimediales Online-Handbuch*) unterschieden werden (siehe Balzert, 2000b, Seite 640 f.).

Beim Lesen eines gedruckten *Benutzungshandbuchs*[Artefakt] ermüden die Augen nicht so schnell, wie beim Lesen des Online-Handbuchs (Balzert, 2000b, Seite 640). So ist beim gedruckten *Benutzungshandbuch*[Artefakt] aus didaktischen Gründen durchaus Redundanz zur Vermittlung der Inhalte erwünscht (Rupietta, 1987, Seite 170). Des Weiteren kann das *Benutzungshandbuch*[Artefakt] zum Nachlesen und zur Vorbereitung auf eine Prüfung verwendet werden. Zur gedruckten Version des *Benutzungshandbuchs*[Artefakt] gehört auch das *Booklet* der CD-ROM. Dieses muss neben allgemeinen Informationen des Herstellers, wie zum Beispiel die Adresse und Hinweise zum Kopierschutz, auch eine kurze Installationsanleitung zum aktuellen *Release* inklusive der Hard- und Software-Anforderungen enthalten. Des Weiteren sollten Hinweise zur Problembehebung und gegebenenfalls die Telefonnummer einer *Hotline* angegeben werden. Um dem Lerner den Einstieg in das virtuelle Labor zu erleichtern und falls sonst kein weiteres gedrucktes *Benutzungshandbuch*[Artefakt] vorliegt, sollte das *Booklet* auch eine kurze Einführung in die Bedienung des virtuellen Labors enthalten (*Schnellstart*).

Ein Online-Handbuch wird wie ein gedrucktes *Benutzungshandbuch*[Artefakt] mit dem Anspruch erstellt, das betreffende Software-Produkt vollständig und umfassend zu beschreiben (Brodbeck und Rupietta, 1994, Seite 216). Daher lassen sich die Verfahrensweisen für den Entwurf und die Gestaltung von gedruckten *Benutzungshandbüchern*[Artefakt] (siehe Boedicker, 1990) hierauf übertragen (Rupietta, 1987, Seite 170). Das Online-Handbuch sollte möglichst der Benutzungsoberfläche und Bedienungslogik des virtuellen Labors folgen, das heißt ebenfalls multimedial gestaltet sein. Des Weiteren sollte es nur die notwendigsten Erklärungen enthalten und aufgabenorientiert verfasst sein. Außerdem sollte es Bedienungsfehlern mit Lösungsvorschlägen begegnen und die *Endanwender*[Rolle] zum aktiven Umgang mit dem virtuellen Labor auffordern und anleiten (siehe Klimsa, 1997, Seite 14). Das Online-Handbuch ermöglicht die Darstellung kontextsensitiver Information im virtuellen Labor. Des Weiteren sollte es die Möglichkeit bieten, beliebige Lesezeichen (*Bookmarks*) zu setzen (Thissen, 2000, Seite 66) und eine Navigations- und Suchfunktion enthalten. Mit der Navigation ist die direkte Auswahl von Informationen zum virtuellen Labor, das heißt zu den Versuchskomponenten und Reaktionsabläufen, möglich. Die Suchfunktion sollte eine Volltext-Suche im Online-Handbuch erlauben und dabei berücksichtigen, dass der Lerner oft nicht genau weiß, was er sucht (siehe Brodbeck und Rupietta, 1994, Seite 218). Das elektronische *Benutzungshandbuch*[Artefakt] wird aus Gründen der schlechteren Lesbarkeit am Bildschirm (siehe oben) jedoch knapper gehalten als ein gedrucktes *Benutzungshandbuch*[Artefakt], ohne dabei den Anspruch nach Vollständigkeit zu verlieren.

Außerdem sollte das *Benutzungshandbuch*[Artefakt] (unabhängig davon ob gedruckt oder elektronisch) mindestens einen Beispieldurchlauf durch das virtuelle Labor enthalten. Dieser beschreibt in linearer Form und Schritt für Schritt, welche Aktivitäten der Lerner im

virtuellen Labor durchzuführen hat. Dabei werden vom Lerner alle wichtigen Bestandteile des virtuellen Labors mindestens einmal durchlaufen, sowie die Benutzungsoberfläche und alle Navigationselemente erklärt und bedient (siehe aufgabenorientierte Gliederung des Benutzungshandbuchs in Balzert, 2000b, Seite 645 bis 648). Ein Beispieldurchlauf durch das virtuelle Labor liegt zum Beispiel der Dokumentation von *GenLab* bei.

7.8.2.2. Erstelle die technische Dokumentation

Die *technische Dokumentation*[Artefakt] wird von den *System-Analytikern*[Rolle] und den *Designern/Entwicklern*[Rolle] erstellt. Im Gegensatz zum *Benutzungshandbuch*[Artefakt] wird die *technische Dokumentation*[Artefakt] nicht an den *Endanwender*[Rolle] ausgeliefert. Die *technische Dokumentation*[Artefakt] setzt sich zusammen aus der *System-Dokumentation* und das *Handbuch für die System-Administration*. Die System-Dokumentation dient zur Wartung des virtuellen Labors und zur Weiterentwicklung des Frameworks bei Neuentwicklungen. Dazu enthält die System-Dokumentation eine detaillierte Beschreibung des Frameworks und dessen Anwendung. Das Handbuch für die System-Administration enthält eine Beschreibung der Installation des virtuellen Labors (inklusive der Quelltexte und der benötigten Standardkomponenten) auf die Entwicklungsumgebung. Außerdem wird die Konfiguration des virtuellen Labors und der verwendeten Standardkomponenten beschrieben. Bei der Erstellung der *technischen Dokumentation*[Artefakt] kann von einer allgemeinen informatischen Vorbildung des Lesers ausgegangen werden. So kann der Inhalt der *technischen Dokumentation*[Artefakt] relativ kurz aber dennoch präzise gehalten werden.

7.8.3. Führe internen Abnahmetest durch

Für den internen Abnahmetest wird von den *Designern/Entwicklern*[Rolle] mit Unterstützung durch den *Konfigurationsmanager*[Rolle] ein *internes Release*[Artefakt] des virtuellen Labors erstellt. Das *interne Release*[Artefakt] setzt sich zusammen aus dem aktuellen *Build* des virtuellen Labors, dem *Benutzungshandbuch*[Artefakt] und der *technischen Dokumentation*[Artefakt]. Die Freigabe des *internen Release*[Artefakt] erfolgt durch den *Projektleiter*[Rolle]. Das *interne Release*[Artefakt] wird dann im Workflow *Test und Evaluation*[Workflow] (siehe Kapitel 7.7) von den *Testern*[Rolle] getestet und von dem *Fachexperten*[Rolle], dem *Fachdidaktiker*[Rolle] und den *Medienspezialisten*[Rolle] evaluiert. Des Weiteren wird im Workflow *Test und Evaluation*[Workflow] das virtuelle Labor von einer kleinen Anzahl ausgewählter *Endanwendern*[Rolle] evaluiert (vergleiche dazu die *formative Evaluation* in Issing, 1997, Seite 213). In den internen Abnahmetests wird das virtuelle Labor außerdem beim *Auftraggeber*[Rolle] präsentiert beziehungsweise demonstriert. Der *Auftraggeber*[Rolle] hat des Weiteren die Möglichkeit das virtuelle Labor selbst zu testen und zu evaluieren. Die möglichst frühzeitige Einbeziehung des *Auftraggebers*[Rolle] und der *Endanwender*[Rolle] in den Test und die Evaluation ist bei virtuellen Laboren besonders wichtig. Nur so lässt sich die Akzeptanz der Benutzungsoberfläche und der Labornavigation seitens des *Auftraggebers*[Rolle] und der *Endanwender*[Rolle] feststellen und aufwendige Änderungen im Nachhinein vermeiden. Der interne Abnahmetest wird bereits mit dem ersten Oberflächenprototypen und anschließend in jeder Iteration des *VirtLab*–Prozess

durchgeführt. Dabei fließen die Ergebnisse und Anregungen aus den Tests und der Evaluation in die weitere Entwicklung des virtuellen Labors mit ein.

7.8.4. Erstelle externes Release

In dieser Aktivität wird von den *Designern/Entwicklern*[Rolle] mit Unterstützung des *Konfigurationsmanagers*[Rolle] ein *externes Release*[Artefakt] des virtuellen Labors erstellt. Dieses wird an den *Auftraggeber*[Rolle] und die *Endanwender*[Rolle] zu Test- und Evaluationszwecken ausgeliefert. Bei Auslieferung an die *Endanwender*[Rolle] setzt sich das *externe Release*[Artefakt] aus dem aktuellen *Build* des virtuellen Labors, dem *Benutzungshandbuch*[Artefakt], dem *Booklet* zur CD-ROM und die Verpackung zusammen. Das *externe Release*[Artefakt] für den *Auftraggeber*[Rolle] enthält außerdem noch die *technische Dokumentation*[Artefakt]. Des Weiteren sind bei der Erstellung des *externen Release*[Artefakt] (gegebenenfalls auch schon für das *interne Release*[Artefakt]) die Quelltexte des virtuellen Labors gegen unbefugte Verwendung zu schützen (Merx, 1999, Seite 128). Die Freigabe des *externen Release*[Artefakt] für den *Auftraggeber*[Rolle] als auch für die *Endanwender*[Rolle] erfolgt durch den *Projektleiter*[Rolle].

7.8.5. Führe externen Abnahmetest durch

Für den externen Abnahmetest (vergleiche *summative Evaluation* in Issing, 1997, Seite 213) sind prinzipiell zwei Varianten möglich, die in der Regel parallel ausgeführt werden. Die erste Variante ist die *Demonstration beim Auftraggeber*. Dazu wird das virtuelle Labor auf den Computer-Systemen des *Auftraggebers*[Rolle] installiert und dort von dem *Projektleiter*[Rolle] demonstriert. Des Weiteren wird das virtuelle Labor vom *Auftraggeber*[Rolle] getestet.

Die zweite Variante sind die β-*Tests*, bei der das virtuelle Labor an eine Vielzahl von *Endanwender*[Rolle] ausgeliefert und getestet wird. Die Koordination der β-Tests erfolgt durch den *Projektleiter*[Rolle]. Umfangreiche β-Tests sind von großer Bedeutung, um verlässliche Aussagen über Installationserfolg, Lauffähigkeit, Ausführungsgeschwindigkeit und Plattformunabhängigkeit des virtuellen Labors machen zu können (vergleiche *Mastering* in Yass, 2000, Seite 150 f.).

Nach der Durchführung der externen Abnahmetests wird eine Liste von Änderungswünschen bezüglich der Anforderungen an das virtuelle Labor erstellt. Diese werden dem *Projektleiter*[Rolle] übergeben und fließen in die weitere Entwicklung des virtuellen Labors mit ein.

7.8.6. Vertreibe das virtuelle Labor

Für den Vertrieb des virtuellen Labors lassen sich prinzipiell drei verschiedene Modelle unterscheiden (vergleiche Nagl u. a., 1999, Seite 105 bis 111). Bei der *individuellen Installation* wird das virtuelle Labor von den *Designern/Entwicklern*[Rolle] auf den Computer-Systemen des *Auftraggebers*[Rolle] installiert. Das ist zum Beispiel dann der Fall, wenn das

virtuelle Labor begleitetend zu einem realen Laborpraktikum an einer Universität eingesetzt werden soll (vergleiche Kronberg, 1998, Seite 30). Nach der Installation des virtuellen Labors führen die *Fachdidaktiker*[Rolle] eine Schulung der Betreuer des Laborpraktikums durch. Außerdem ist eine regelmäßige Wartung der Computer-Systeme notwendig. Diese kann in einem Wartungsvertrag mit dem *Auftraggeber*[Rolle] geregelt werden.

Bei der *Massenproduktion* wird das virtuelle Labor auf einer CD-ROM vertrieben. Nach Abschluss der β-Tests (siehe *Führe externen Abnahmetest durch*[Aktivität] in Kapitel 7.8.5) wird eine *Master-CD* erstellt und vervielfältigt. Diese Aktivität ist von einem *Designer/Entwickler*[Rolle] durchzuführen (siehe Yass, 2000, Seite 150). Gerade bei einer Massenproduktion ist eine umfangreiche und sorgfältige β-Testphase von großer Bedeutung, da sich Fehler auf der Master-CD auf alle Kopien auswirken. Es ist außerdem auch denkbar, dass das virtuelle Labor den *Endanwendern*[Rolle] als Download aus dem Internet zur Offline-Nutzung zur Verfügung gestellt wird. Dann zum Beispiel als Evaluationsversion mit eingeschränkter Funktionalität.

Die dritte Möglichkeit ist die *Online-Nutzung*. Das virtuelle Labor wird von den *Endanwendern*[Rolle] direkt über das Internet gestartet und als Client/Server-Anwendung genutzt. Die Nutzung kann dann zum Beispiel direkt über das Internet mit dem *Endanwender*[Rolle] abgerechnet werden. Des Weiteren benötigen eventuelle Änderungen am virtuellen Labor bei der Online-Nutzung keine separate Aktualisierung mehr. Die Änderungen an dem virtuellen Labor werden automatisch beim nächsten Start des Clients nachgezogen.

Des Weiteren ist bezüglich des Vertriebs des virtuellen Labors und das dynamische Laden von Versuchen zu überlegen, ob weitere (eventuell kostenpflichtige) Erweiterungen zum virtuellen Labor angeboten werden sollen. Diese Erweiterungen bestehen zum Beispiel aus zusätzlichen Versuchen oder Fertigkeiten, die gegebenenfalls auch neue Laborgeräte, Behälter und Substanzen enthalten können. Die Erweiterungen werden entweder über CD-ROM vertrieben oder im Internet zur Verfügung gestellt.

Bei dem Vertrieb des virtuellen Labors ist außerdem zu berücksichtigen, ob eine Installation des virtuellen Labors (oder Teilen davon) auf den Computer-Systemen der *Endanwender*[Rolle] notwendig ist. Ist dies der Fall, so muss neben der Installation auch eine Deinstallation des virtuellen Labors im Lieferumfang enthalten sein.

7.9. Konfigurationsmanagement

Die Struktur des Workflow zum *Konfigurationsmanagement*[Workflow] ist identisch mit dem des Rational Unified Process (siehe Kruchten, 2000, Seite 216). Gegenstand des Konfigurationsmanagement ist die Verwaltung und Komposition des virtuellen Labors aus einer Menge von Komponenten. Zur Verwaltung werden entsprechende Werkzeuge des Versionsmanagements eingesetzt, die die Änderungen an den Komponenten während der Entwicklung oder der Wartung des virtuellen Labors protokollieren (Nagl u. a., 1999, Seite 169).

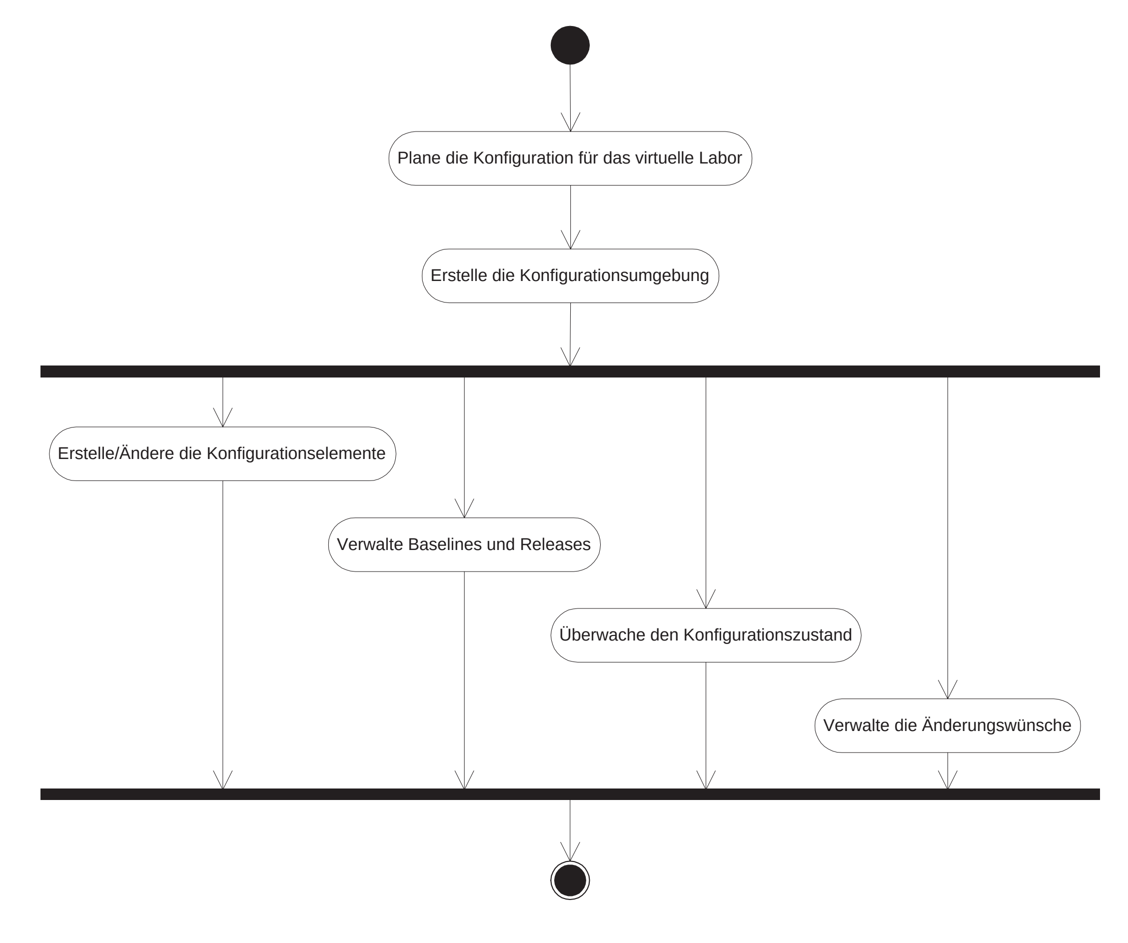


Abbildung 7.28.: Der Workflow für das Konfigurationsmanagement

Zunächst wird die Konfigurationsumgebung zur Entwicklung des virtuellen Labors entworfen und erstellt. Danach teilt sich das *Konfigurationsmanagement*^{Workflow} in vier parallele Aktivitäten. Das ist zum einen die Erstellung und Änderung von Elementen der Konfigurationsumgebung. Des Weiteren werden die *Baselines* und *Releases* zum virtuellen Labor verwaltet. Gleichzeitig wird der Konfigurationszustand während der Durchführung der Iteration überwacht und eventuelle Änderungswünsche an den Anforderungen des virtuellen Labors verwaltet.

Die Aktivitäten des *Konfigurationsmanagements*^{Workflow} werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Plane die Konfiguration für das virtuelle Labor*^{Aktivität} (siehe Kapitel 7.9.1),
- *Erstelle die Konfigurationsumgebung*^{Aktivität} (siehe Kapitel 7.9.2),
- *Erstelle/Ändere die Konfigurationselemente*^{Aktivität} (siehe Kapitel 7.9.3),
- *Verwalte Baselines und Releases*^{Aktivität} (siehe Kapitel 7.9.4),
- *Überwache den Konfigurationszustand*^{Aktivität} (siehe Kapitel 7.9.5) und

- *Verwalte die Änderungswünsche*[Aktivität] (siehe Kapitel 7.9.6).

7.9.1. Plane die Konfiguration für das virtuelle Labor

In dieser Aktivität des *Konfigurationsmanagements*[Workflow] wird vom *Konfigurationsmanager*[Rolle] der *Konfigurationsmanagementplan*[Artefakt] erstellt. Dieser beinhaltet das Vorgehen zur Verwaltung der *Repositories* und *Baselines* des virtuellen Labors, sowie der Arbeitsbereiche der einzelnen *Projektmitarbeiter*[Rolle]. Dazu wird bestimmt, welche Versionsmanagementsysteme für die Entwicklung des virtuellen Labors eingesetzt werden. Allgemein sind sämtliche Quelltexte, Medien und Dokumente des virtuellen Labors mit Hilfe eines entsprechenden Versionsmanagementsystems zu verwalten.

Des Weiteren wird eine Strategie für die Datensicherung erstellt (*Backup-Strategie*). Die Backup-Strategie legt fest, wann welche Daten der Entwicklungsumgebung gesichert werden (siehe Yass, 2000, Seite 125). Außerdem werden separat zu den regelmäßigen Sicherheitskopien der Entwicklungsumgebung die einzelnen *Builds* des virtuellen Labors mit Datum und Uhrzeit versehen und aufbewahrt, sobald eine bestimmte *Baseline* oder einen Meilenstein erreicht wurde. Dadurch wird ein einfacher Zugriff auf ältere und eventuell stabilere Versionen des virtuellen Labors ermöglicht. Außerdem kann durch einen direkten Vergleich zwischen zwei Versionen des virtuellen Labors der Projektfortschritt einfacher beurteilt werden. Zum Abschluss der Aktivität wird der *Konfigurationsmanagementplan*[Artefakt] vom *Projektleiter*[Rolle] bestätigt.

7.9.2. Erstelle die Konfigurationsumgebung

Der *Konfigurationsmanager*[Rolle] und der *System-Administrator*[Rolle] erstellen auf der Grundlage des *Konfigurationsmanagementplans*[Artefakt] und des *Implementierungsplans*[Artefakt] die Konfigurationsumgebung für die Entwicklung des virtuellen Labors. Dazu werden die benötigten *Repositories* für das Versionsmanagement eingerichtet und entsprechende Arbeitsbereiche für die einzelnen *Projektmitarbeiter*[Rolle] zur Verfügung gestellt. Die Konfigurationselemente des virtuellen Labors sind die Quelltexte, Medien und Dokumente. Zur Verwaltung der Quelltexte kann zum Beispiel das *Concurrent Versions System* (CVS, siehe: http://www.cvshome.org/) verwendet werden. Wird für die Entwicklung des virtuellen Labors das Autorensystem *Director* eingesetzt, so kann dieses zur Verwaltung der Medien verwendet werden. *Director* erlaubt jedoch nur ein rudimentäres Management der Medien, so können zum Beispiel keine verschiedenen Versionen eines Mediums verwaltet werden. Zur Verwaltung der Dokumente des virtuellen Labors kann insbesondere in Hinblick auf die einfache Bedienbarkeit des Werkzeuges zum Beispiel das *Basic Support for Cooperative Work* (BSCW, siehe: http://bscw.gmd.de/) eingesetzt werden.

7.9.3. Erstelle/Ändere die Konfigurationselemente

Alle Elemente der Konfigurationsumgebung, die ein *Projektmitarbeiter*[Rolle] erstellt oder ändert, werden nach erreichen einer bestimmten *Baseline* in die *Repositories* eingespielt. In bestimmten Abständen, zum Beispiel zu Beginn des Arbeitstages oder nach Absprache, werden dann die Änderungen an den *Repositories* auf die Arbeitsbereiche der einzelnen *Projektmitarbeiter*[Rolle] übertragen.

7.9.4. Verwalte Baselines und Releases

Der *Konfigurationsmanager*[Rolle] entscheidet nach dem *Konfigurationsmanagementplan*[Artefakt], ob die Entwicklung des virtuellen Labors eine bestimmte *Baseline* erreicht hat. Wurde eine *Baseline* erreicht, so wird ein *Release* des virtuellen Labors erstellt. Dieses kann entweder ein *internes Release*[Artefakt] oder ein *externes Release*[Artefakt] sein (siehe *Einsatz*[Workflow] in Kapitel 7.8).

7.9.5. Überwache den Konfigurationszustand

Der *Konfigurationsmanager*[Rolle] prüft anhand des *Konfigurationsmanagementplans*[Artefakt], ob die technischen Anforderungen an das virtuelle Labor eingehalten werden. Des Weiteren kontrolliert der *Konfigurationsmanager*[Rolle], ob alle Artefakte in einer geordneten Form in den *Repositories* gespeichert und verwaltet werden und prüft, ob alle Artefakte vorhanden und gültig sind. Außerdem muss der *Konfigurationsmanager*[Rolle] sicherstellen, dass alle *Projektmitarbeiter*[Rolle] auf die benötigten Artefakte und *Baselines* zugreifen können.

7.9.6. Verwalte die Änderungswünsche

Alle von den *Projektmitarbeitern*[Rolle] geäußerten Änderungswünsche an das virtuelle Labor werden vom *Konfigurationsmanager*[Rolle] gesammelt. Dieser strukturiert und klassifiziert die Änderungswünsche und überprüft die Aussagen anhand des aktuellen *Build* des virtuellen Labors. Der *Projektleiter*[Rolle] entscheidet dann über jeden Änderungswunsch, ob er akzeptiert oder abgelehnt wird. Gegebenenfalls wird dazu eine Absprache mit dem *Auftraggeber*[Rolle] vorgenommen. Außerdem werden alle beteiligten *Projektmitarbeiter*[Rolle] informiert, damit diese die Änderungen argumentativ und inhaltlich nachvollziehen können.

7.10. Entwicklungsumgebung

Aufgabe des Workflows zur *Entwicklungsumgebung*[Workflow] ist es, den *Projektmitarbeitern*[Rolle] eine geeignete Entwicklungsumgebung für die Erstellung des virtuellen Labors zur Verfügung zu stellen und während des Entwicklungsprozesses zu betreuen.

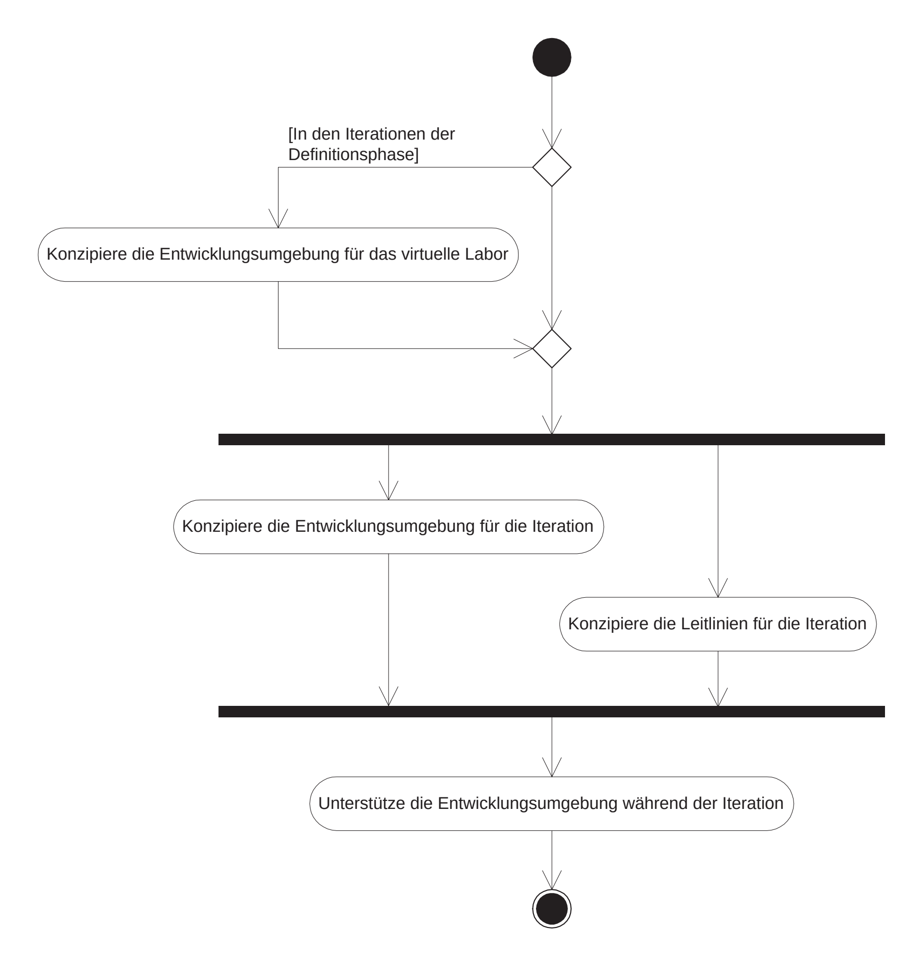


Abbildung 7.29.: Der Workflow für die Entwicklungsumgebung

Die Struktur des Workflow zur *Entwicklungsumgebung*^Workflow entspricht der des entsprechenden Workflows des Rational Unified Process (siehe Kruchten, 2000, Seite 224) und beginnt in der Definitionsphase mit der Konzeption der Entwicklungsumgebung für das virtuelle Labor. Für jede Iteration des *VirtLab*–Prozesses wird dann eine konkrete Entwicklungsumgebung konzipiert und entsprechende Leitlinien erstellt. Während der Iteration stehen diese Leitlinien dann den *Projektmitarbeitern*^Rolle zur Durchführung ihrer Aktivitäten zur Verfügung.

Die Aktivitäten der *Entwicklungsumgebung*^Workflow werden in den folgenden Kapiteln im Detail beschrieben. Im Einzelnen sind das:

- *Konzipiere die Entwicklungsumgebung für das Projekt*^Aktivität (siehe Kapitel 7.10.1),
- *Konzipiere die Entwicklungsumgebung für die Iteration*^Aktivität (siehe Kapitel 7.10.2),
- *Konzipiere die Leitlinien für die Iteration*^Aktivität (siehe Kapitel 7.10.3) und
- *Unterstütze die Entwicklungsumgebung während der Iteration*^Aktivität (siehe Kapitel 7.10.4).

7.10.1. Konzipiere die Entwicklungsumgebung für das Projekt

In diesem Teil des Workflows der *Entwicklungsumgebung*[Workflow] wird die Arbeitsumgebung für die Entwicklung des virtuellen Labors festgelegt. Diese Aktivität wird von dem *Konfigurationsmanager*[Rolle] unter Rücksprache mit dem *Projektleiter*[Rolle] durchgeführt. Die Festlegung der Entwicklungsumgebung lässt sich nach folgenden vier Kriterien unterscheiden:

- *Software*: Es ist festzulegen, auf welchem Betriebssystem, mit welcher Programmiersprache und mit welcher Entwicklungsumgebung das virtuelle Labor erstellt wird. In diesem Zusammenhang ist auch zu prüfen, ob ein Autorensystem als Entwicklungsumgebung eingesetzt wird. Des Weiteren ist zu prüfen, ob spezielle Werkzeuge verwendet werden können. Gegebenenfalls ist auch eine geeignete Datenbank und ein Expertensystem auszuwählen.

- *Hardware*: Es ist zu bestimmen, auf welchen Computer-Systemen das virtuelle Labor entwickelt werden soll beziehungsweise welche Entwicklungsplattformen (*CASE-Plattform*) dafür zur Verfügung stehen. Gegebenenfalls ist außerdem die benötigte Hardware zur Einbindung externer Geräte, die an das virtuelle Labor angeschlossen werden, zu bestimmen.

- *Orgware*: In diesem Zusammenhang ist zu prüfen, welche Netztopologie für die Entwicklung des virtuellen Labors benötigt wird. Gegebenenfalls ist ein Zugang zum Internet zur Verfügung zu stellen. Außerdem ist festzulegen, ob Tele-Arbeit möglich oder wünschenswert ist und welche zusätzlichen Hilfsmittel dazu eingesetzt werden, wie zum Beispiel Video-Konferenzen.

- *Teachware*: Es sind sämtliche Informationsquellen wie zum Beispiel Bücher und Experten zu bestimmen, die für die Entwicklung des virtuellen Labors zur Verfügung stehen. Außerdem ist festzulegen, ob Seminare abgehalten oder Tagungen besucht werden sollen. Dazu sind gegebenenfalls Reisen notwendig, bei denen zusätzliche Kosten anfallen.

Des Weiteren wird die Spezifikationssprache zur Spezifikation der Lerneinheiten festgelegt, sowie weitere Schablonen und Richtlinien für die Entwicklung virtueller Labore identifiziert und entwickelt. Außerdem sind Entwicklungsfälle in skizzierter Form zu erstellen und eine Bewertung der Entwicklungsorganisation vorzunehmen (Kruchten, 2000).

7.10.2. Konzipiere die Entwicklungsumgebung für die Iteration

In dieser Aktivität legt der *Konfigurationsmanager*[Rolle] unter Beratung mit dem *System-Administrator*[Rolle] die konkrete Entwicklungsumgebung für die anstehende Iteration fest. Dazu gehört die Festlegung der verwendeten Entwicklungswerkzeuge und die Bereitstellung von Schablonen und Richtlinien für die Iteration.

7.10.3. Konzipiere die Leitlinien für die Iteration

In dieser Aktivität legt der *Konfigurationsmanager*[Rolle] die Leitlinien für die aktuelle Iteration fest. Die konkrete Ausarbeitung der Leitlinien wird ebenfalls von dem *Konfigurationsmanager*[Rolle] durchgeführt. Unter Absprache mit dem *Projektleiter*[Rolle] und dem Einverständnis aller *Projektmitarbeiter*[Rolle] werden die Leitlinien verabschiedet. Diese sind für alle *Projektmitarbeiter*[Rolle] verpflichtend. Für die Entwicklung virtueller Labore sind zum Beispiel *Richtlinien für die Dokumentation*[Artefakt], *Werkzeugrichtlinien*[Artefakt] und *Programmierrichtlinien*[Artefakt] sinnvoll. Außerdem kümmert sich der *Konfigurationsmanager*[Rolle] um die Einführung der Leitlinien in den Entwicklungsprozess und überwacht deren Einhaltung während der Durchführung der Iteration.

7.10.4. Unterstütze die Entwicklungsumgebung während der Iteration

Der *System-Administrator*[Rolle] hilft den *Projektmitarbeitern*[Rolle] bei technischen Problemen bezüglich der verwendeten Entwicklungsplattform. Dies betrifft neben dem Betriebssystem auch die eingesetzten Entwicklungswerkzeuge. Die Unterstützung der Entwicklungsumgebung durch den *System-Administrator*[Rolle] erfolgt während jeder Iteration des Software-Entwicklungsprozesses virtueller Labore.

8. Bewertung und Ausblick

Der in dieser Arbeit vorgestellte *VirtLab*–Prozess stellt einen systematischen Ansatz eines Vorgehensmodells und der zugehörenden Entwicklungsmethodik für virtuelle Labore dar. Als Grundlage des *VirtLab*–Prozesses wurde der um den Workflow der Geschäftsprozessmodellierung reduzierte Rational Unified Process verwendet. Dieser wurde um einen Workflow für die Medienproduktion und einen für das Tutorkonzept erweitert (siehe Kapitel 3.5).

8.1. Bewertung

Zur Anpassung des Rational Unified Process an den Anwendungsfall virtuelle Labore mussten die vorhandenen Workflows des Rational Unified Process nur relativ wenig verändert werden. Dadurch wird ein hoher Wiedererkennungseffekt bei denjenigen Lesern erzeugt, die den Rational Unified Process bereits kennen, was wiederum die Akzeptanz des *VirtLab*–Prozess erhöht. Nach den Bezeichnern der Aktivitäten in den einzelnen Workflows des *VirtLab*–Prozess zu urteilen, könnte man zunächst annehmen, dass es sich bei dem *VirtLab*–Prozess um ein allgemeines Vorgehensmodell zur Entwicklung von multimedialen Lehr- und Lernsystemen handelt. Vielmehr konnten jedoch die durchzuführenden Aktivitäten sowie die zu erstellenden und verwendenden Artefakte bei der Entwicklung virtueller Labore in die vorhandenen Workflows des Rational Unified Process eingebettet werden. Die Inhalte der Aktivitäten des *VirtLab*–Prozess sind speziell auf den Anwendungsfall virtuelle Labore zugeschnitten, wie aus den jeweiligen Beschreibungen und graphischen Darstellungen der Aktivitäten zu entnehmen ist. Des Weiteren wird durch den *VirtLab*–Prozess eine bestimmte *Architektur*[Artefakt] des Frameworks für virtuelle Labore unterstellt. Damit ist der *VirtLab*–Prozess ein architekturspezifisches Prozessmodell, wie auch der Object Engineering Process.

Von dem *VirtLab*–Prozess werden sämtliche zu Beginn des Kapitels 3.1 genannten Aspekte virtueller Labore berücksichtigt. Die Grundlage zur Unterstützung der Heterogenität des Entwicklerteams stellt auf der Ebene des Vorgehensmodells das Rollenkonzept dar. Dazu sind die an der Entwicklung des virtuellen Labors beteiligten Rollen des *VirtLab*–Prozess in Kapitel 5 detailliert aufgeschlüsselt und beschrieben worden. Des Weiteren existiert eine eigene Aktivität im *VirtLab*–Prozess zur Berücksichtigung der verschiedenen Fachsprachen und Denkweisen der *Projektmitarbeiter*[Rolle] (siehe *Berücksichtige Sprachgebrauch und Denkweisen der Projektmitarbeiter*[Aktivität] in Abschnitt 7.3.2.2). Auf der Ebene der Entwicklungsmethodik wird die konkrete Zusammenarbeit des heterogenen Entwicklerteams,

das heißt der einzelnen Rollen mit Hilfe von Checklisten zu den Artefakten des *VirtLab*–Prozess beschrieben (siehe Anhang B).

Die Einbeziehung des *Auftraggebers*^{Rolle} in die Entwicklung des virtuellen Labors erfolgt während der Erstellung des *Pflichtenheftes*^{Artefakt} und in jeder Iteration des *VirtLab*–Prozesses bei der Evaluation des virtuellen Labors im Workflow *Test und Evaluation*^{Workflow}. Diese wird in der Aktivität *Führe internen Abnahmetest durch*^{Aktivität} (siehe Kapitel 7.8.3) initiiert. Ebenso wie die Einbeziehung des *Auftraggebers*^{Rolle} erfolgt die Einbeziehung der *Endanwender*^{Rolle} in die Evaluation des virtuellen Labors während jeder Iteration des *VirtLab*–Prozesses. Des Weiteren sind im *VirtLab*–Prozess auch β-Tests vorgesehen, die von den *Endanwendern*^{Rolle} durchgeführt werden.

Die multimedialen Aspekte virtueller Labore werden vom *VirtLab*–Prozess in einem eigenen Workflow der *Medienproduktion*^{Workflow} berücksichtigt. Hervorzuheben ist dabei die detaillierte Aufschlüsselung der Medientypen und möglichen Einsatzarten der Medien. Des Weiteren wird das allgemeine Layout und die Gestaltung der Benutzungsoberfläche im Workflow der *Medienproduktion*^{Workflow} in der Aktivität *Erstelle das Gesamtdesign*^{Aktivität} (siehe Kapitel 7.5.1) berücksichtigt.

Die didaktischen Aspekte virtueller Labore werden im Workflow des *Tutorkonzeptes*^{Workflow} mit der Erstellung eines umfassenden *didaktischen Konzeptes*^{Artefakt} berücksichtigt. Dabei wird insbesondere auf die Lerneranalyse, Festlegung von Navigationsstruktur und -möglichkeiten und die Spezifikation von Aufbau und Ablauf der Versuche und Fertigkeiten eingegangen. Dies unterscheidet den *VirtLab*–Prozess von den Vorgehensmodellen nach (Yass, 2000) und (Nagl u. a., 1999), deren didaktische Überlegungen auf die Entwicklung klassischer Lehr- und Lernsysteme ausgerichtet sind.

Da der *VirtLab*–Prozess auf den Rational Unified Process aufbaut und mit Hilfe von Workflows definiert und beschrieben wird, kann auch der *VirtLab*–Prozess relativ einfach erweitert und verfeinert werden. So können zum Beispiel die bisher als optional gekennzeichneten Aktivitäten ausgebaut werden. Oder es kann die Beschreibung der Aktivitäten, die noch nicht im Detail beleuchtet wurden, verfeinert werden.

Die Wiederverwendbarkeit bestehender Entwicklungen steht beim *VirtLab*–Prozess im Vordergrund, insbesondere um die Entwicklungskosten virtueller Labore zu senken. So ist bei der Definition der *Architektur*^{Artefakt} die Wiederverwendbarkeit des Frameworks zu prüfen. Da das virtuelle Labor auf Komponenten basierend entworfen und implementiert wird, ist eine hohe Wiederverwendbarkeit von bereits realisierten Arbeitsflächen, Laborgeräten, Behältern und Substanzen sichergestellt. Außerdem wird die Wiederverwendung von bereits vorhandenen Medien in einer eigenen Aktivität der *Medienproduktion*^{Workflow} berücksichtigt (siehe *Analysiere vorhandene Medien*^{Aktivität} in Kapitel 7.5.3).

Im Gegensatz zum Rational Unified Process erfolgt im *VirtLab*–Prozess eine Aufteilung in Test und Evaluation des virtuellen Labors. Damit werden neben der Prüfung der funktionalen Anforderungen auch die gestalterischen, didaktischen und fachlichen Aspekte des virtuellen Labors berücksichtigt, was dem Wunsch nach einer verbesserten Qualitätssicherung multimedialer Lehr- und Lernsysteme nachkommt (Nagl u. a., 1999, Seite 104). Die Tests des virtuellen Labors sind insofern als angemessen zu beurteilen, als dass sie in jeder Iteration des

VirtLab–Prozesses durchgeführt werden. Die Evaluation des virtuellen Labors ist durch die frühzeitige und regelmäßige Einbeziehung des *Auftraggebers*^Rolle und der *Endanwender*^Rolle in den *VirtLab*–Prozess ebenfalls als angemessen zu beurteilen.

Die erste Risikoanalyse zur Entwicklung des virtuellen Labors besteht aus der Schätzung von Umfang und Risiko der Entwicklung (inklusive *Vorstudien*^Artefakt) und der Erstellung eines initialen *Gesamtentwicklungsplans*^Artefakt. Diese umfassende Risikoanalyse wird als angemessen bewertet, weil es zu Beginn der Entwicklung besonders wichtig ist, die Risiken zu identifizieren und einzuschätzen. Des Weiteren wird am Ende jeder Iteration in einer eigenen Aktivität des *Projektmanagements*^Workflow der restliche Umfang und das verbleibende Risiko zur Entwicklung des virtuellen Labors geschätzt (siehe *Schätze restlichen Umfang und Risiko des Projekts*^Aktivität in Kapitel 7.1.10). Der Aufwand für die Risikoanalyse nach jeder Iteration wird als angemessen eingestuft, da die Erstellung der *Vorstudien*^Artefakt wegfällt und die Risikoanalyse im Wesentlichen aus der Aktualisierung und Verfeinerung des *Gesamtentwicklungsplans*^Artefakt besteht.

Insgesamt lässt sich festhalten, dass der *VirtLab*–Prozess dem Wunsch nach verbesserten Prozessmodellen für die Entwicklung von multimedialen Lehr- und Lernsystemen (siehe Nagl u. a., 1999, Seite 104) bezüglich des Anwendungsfalls virtuelle Labore gerecht wird. Mögliche Ergänzungen und Erweiterungen zukünftiger Arbeiten werden im folgenden Kapitel 8.2 identifiziert.

8.2. Ausblick

Der in dieser Arbeit beschriebene *VirtLab*–Prozess ist wie alle anderen Prozessmodelle mit Sicherheit nicht vollständig und erhebt auch keinen Anspruch auf Vollständigkeit (vergleiche Ambler, 2001, Seite 24). Mögliche Erweiterungen und Ergänzungen zum *VirtLab*–Prozess sind:

- Ergänzen des *VirtLab*–Prozesses um die Anbindung weiterer Ein- und Ausgabegeräte, wie zum Beispiel die Anbindung von *Handgloves*, einer *Shutter-Brille* oder die Darstellung des virtuellen Labors mit Hilfe einer *3D-Workbench* (weitere siehe Boles, 1994, Seite 40). Des Weiteren ist es auch denkbar eine zweite Computer-Maus an das virtuelle Labor anzuschließen, um beide Hände eines realen Laborpraktikanten im virtuellen Labor nachbilden zu können.

- Stärkere Berücksichtigung der Evolutionsphase des virtuellen Labors. Dies ist dann sinnvoll, wenn mit dem *Auftraggeber*^Rolle, wie zum Beispiel eine Universität, ein Wartungsvertrag abgeschlossen wird. In der Evolutionsphase ist dann zum Beispiel zu überlegen, wie eine neue Version des virtuellen Labors in die Einsatzumgebung aufgespielt werden kann (vergleiche Ambler, 2001, Seite 20).

- Ergänzen des *VirtLab*–Prozess um einen Workflow zum Infrastrukturmanagement. Dies ist dann wichtig, wenn in dem Unternehmen mehrere virtuelle Labore gleichzeitig

entwickelt werden sollen. Das Infrastrukturmanagement regelt die gemeinsame Verwendung von Ressourcen der Entwicklungsumgebung und insbesondere der *Repositories* mit den darin enthaltenden Quelltexten, Medien und Dokumenten. In diesem Zusammenhang ist auch zu überlegen, wie der *VirtLab*–Prozess mit seinen Workflows, Rollen, Aktivitäten und Artefakten in den Gesamtkontext eines Workflow-Managementsystems des Unternehmens eingebunden werden kann.

- Die Qualität des *VirtLab*–Prozess bewerten und kontinuierlich verbessern. Dabei stehen eine Reihe von Vorgehensweisen zur Bewertung und Verbesserung von Prozessmodellen für die Software-Entwicklung zur Verfügung, wie zum Beispiel CMM, SPICE (=ISO 15504), ISO 9001 und BOOTSTRAP. Zur Evaluierung des *VirtLab*–Prozesses ist es zum Beispiel denkbar, ein Pilotprojekt durchzuführen und dabei insbesondere die Zusammenarbeit des heterogenen Entwicklerteams zu analysieren und festzuhalten.

- Erstellen von Prozesssichten auf den *VirtLab*–Prozess. Bei einer Prozesssicht werden die Aktivitäten zu einem Thema, zu einem bestimmten Aspekt oder für eine bestimmte Rolle zusammengefasst. Prozesssichten sind daher prinzipiell unvollständige Ausschnitte aus dem Vorgehensmodell. Darüber hinaus werden Prozesssichten auch unabhängig von der Einteilung in Phasen und Workflows beschrieben (aus oose.de GmbH, 1999). Beispiel einer solchen Prozesssicht ist die pragmatische Sicht auf den *VirtLab*–Prozess, die in Anhang C zu finden ist. Eine weitere Prozesssicht ist die Beschreibung des *VirtLab*–Prozess durch einen einzigen zusammenhängenden Workflow, in dem die einzelnen in Kapitel 7 beschriebenen Workflows miteinander verzahnt werden. Diese Prozesssicht ist gleichzeitig auch eine alternative Darstellung des *VirtLab*–Prozess im Gegensatz zur Abbildung 4.1.

Neben diesen Erweiterungen und Ergänzungen sind auch eine Reihe von Entwicklungswerkzeugen denkbar, um die Entwicklungsmethodik des *VirtLab*–Prozess zu unterstützen. So können zum Beispiel folgende Werkzeuge realisiert werden (vergleiche dazu die Werkzeuge in Aden u. a., 1999, Seite 41 und 42):

- *Werkzeug zur Anpassung des Metaobjektmodells*: Das Werkzeug basiert auf dem Metaobjektmodell für virtuelle Labore und unterstützt die Definition konkreter virtueller Laborgeräte, Behälter und Substanzen sowie deren Attribute und Methoden (siehe *Verwende das Metaobjektmodell für virtuelle Labore*^{Aktivität} in Abschnitt 7.3.2.1).

- *Werkzeug zur Erstellung des Simulationsmodells*: Während mit dem Werkzeug zur Anpassung des Metaobjektmodells die einzelnen Versuchskomponenten definiert werden können, kann mit diesem Werkzeug das Zusammenspiel und die Interdependenzen der einzelnen Laborgeräte, Behälter und Substanzen im virtuellen Labor festgelegt werden (siehe Abschnitt 7.3.2.1).

- *Werkzeug zur Spezifikation von Aufbau und Ablauf der Lerneinheiten*: Dieses Werkzeug lässt sich in zwei Aufgabenbereiche zerlegen. Zum einen kann der Aufbau der

Versuche und Fertigkeiten mit diesem Werkzeug spezifiziert werden, das heißt es wird bestimmt, auf welchen Arbeitsflächen sich welche Versuchskomponenten befinden sollen. Des Weiteren kann mit dem Werkzeug der Ablauf der Versuche und Fertigkeiten bestimmt werden, das heißt es werden die einzelnen Arbeitsschritte festgelegt, die zur erfolgreichen Durchführung der Lerneinheiten zu erledigen sind (siehe *Spezifiziere Aufbau und Ablauf des Versuchs bzw. der Fertigkeit*[Aktivität] Abschnitt 7.3.4.1).

- *Werkzeug zur Erstellung des Drehbuchs*[Artefakt]*:* Es ist denkbar, die Erstellung des *Drehbuchs*[Artefakt] mit Hilfe eines Werkzeuges geeignet zu unterstützen. Das Werkzeug kann zum Beispiel bestimmte Felder der Drehbuchseiten automatisch ausfüllen und zur Verwaltung der Drehbuchseiten verwendet werden. Die Skizzierung von Aufbau und Ablauf der einzelnen Lerneinheiten erfolgt dabei zum Beispiel mit Hilfe eines Graphiktabletts. Dafür können gegebenenfalls auch Miniaturansichten zu den Versuchskomponenten erstellt und verwendet werden. Das Werkzeug hat gegenüber dem handskizzierten *Drehbuch*[Artefakt] zum Beispiel den Vorteil, dass auf einfache Weise Änderungen und Anpassungen am Versuchsaufbau und -ablauf durchgeführt werden können (siehe *Spezifiziere Aufbau und Ablauf des Versuchs bzw. der Fertigkeit*[Aktivität] in Abschnitt 7.3.4.1).

- *Werkzeug zur Integration des Hintergrundwissens*: Mit Hilfe dieses Werkzeuges können die einzelnen Medien zum Hintergrundwissen der Lerneinheiten in das virtuelle Labor integriert werden. Auch besteht die Möglichkeit, Texte zu editieren und Querverweise einzufügen. Das Werkzeug zur Integration des Hintergrundwissens wird von den *Fachexperten*[Rolle] und den *Fachdidaktikern*[Rolle] bedient (siehe *Spezifiziere das Hintergrundwissen zur Lerneinheit* in Kapitel 7.3.5).

- *Werkzeug zur Integration der Medien*: Mit Hilfe dieses Werkzeuges können von den *Medienspezialisten*[Rolle] einzelne Medien der Arbeitsflächen und Versuchskomponenten in das virtuelle Labor integriert werden. So kann zum Beispiel bei der 2D-Darstellung des virtuellen Labors das Werkzeug zum Zusammenfügen von einzelnen Medien zu einem Abbild eines Laborgerätes oder Behälters verwendet werden. Des Weiteren kann mit dem Werkzeug zu jedem Zustand des Laborgerätes ein entsprechendes Abbild (Visualisierung) zugeordnet werden (siehe *Integriere die Medien*[Aktivität] in Kapitel 7.6.4).

- *Werkzeug zur Wartung und Pflege des VirtLab–Prozess*: Auch zur Wartung und Pflege des *VirtLab*–Prozess selbst ist ein Werkzeug denkbar. Als Grundlage dafür kann zum Beispiel eine Datenbank eingesetzt werden. Mit Hilfe des Werkzeuges können dann die Inhalte der Datenbank erstellt, geändert und gelöscht werden. Zur Darstellung des *VirtLab*–Prozess kann eine auf *Hypertext* basierende Benutzungsoberfläche verwendet werden, wie zum Beispiel bereits für den Rational Unified Process und den Object Engineering Process realisiert. Die einzelnen Seiten der Benutzungsoberfläche werden dann automatisch aus den Inhalten der Datenbank generiert. Zur Pflege des *VirtLab*–Prozess kann aber auch ein bereits existierendes *Content Management System* eingesetzt werden, wie zum Beispiel das Redaktionssystem *RedDot* der Firma *RedDot Solutions AG* (siehe http://www.reddot.de/).

Abschließend bleibt noch die Möglichkeit zu betrachten, den *VirtLab*–Prozess auf Software-Systeme zu erweitern, deren wesentlichen Merkmale identisch mit den hier definierten virtuellen Laboren sind. Als wesentliche Merkmale werden dabei die Modellierung einer bestimmten Domäne auf ein Computer-System, die hohe Interaktivität, der didaktische Anspruch und die Medienproduktion verstanden. Die Erweiterung auf derartige Software-Systeme lässt sich aufgrund der guten Adaptierbarkeit des *VirtLab*–Prozess (siehe Kapitel 8.1) relativ einfach durchführen. Ein Beispiel eines solchen Software-Systems ist das am OFFIS in der Abteilung Lehr- und Lernsysteme entwickelte *Wissensbasierte, Interaktive Krankenzimmer - Einrichtungs - Ausbildungssystem* (WI\KEA, siehe: http://lls.informatik.uni-oldenburg.de/lls/Projekte/Ikea/default.html).

Anhang

A. Beispiel eines Anwendungsfalls

Ausschnitt aus dem von *Fachexperten*Rolle erstellten Versuchsprotokoll zur Gelelektrophorese (aus *GenLab*):

- $1g$ Agarose in dem Messbecher auf der zuvor tarierten Waage abwiegen.
- $100ml$ 1x Elektrophoresepuffer (TBE- oder TAE-Puffer) abmessen und zur Agarose in den Messbecher geben.
- Durch Schütteln die Agarose im Puffer suspendieren.
- Durch kurzes Aufkochen der Suspension ($ca.120s$) in der Mikrowelle die Agarose im Puffer in Lösung bringen.
- Die Agaroselösung auf etwa $60°C$ abkühlen lassen.
- $6\mu l$ Ethidiumbromid-Lösung ($10mg/ml$) zur Agaroselösung hinzupipettieren, so dass eine Ethidiumbromid-Endkonzentration im Gel von $0,6\mu g/ml$ erreicht wird.
- Die aufgekochte Lösung durch sachtes Schwenken des Messbechers auf eine einheitliche Konzentration bringen.
- Den Probenkamm in den Gelträger stecken.
- Die auf etwa $60°C$ abgekühlte Agaroselösung auf den Gelträger gießen.
- Nach dem Erstarren der Agaroselösung den Gelkamm vorsichtig aus dem Gel herausziehen.
- Die Gelelektrophoresekammer mit TBE-Puffer füllen.
- Das abgekühlte Gel zusammen mit dem Gelträger aus der Gelkammer herausnehmen und in die Elektrophoresekammer legen.
- Jeweils den gesamten Reaktionsansatz ($25\mu l$) in eine Geltasche pipettieren.
- In die diesen drei Geltaschen benachbarte Tasche $15\mu l$ der DNA-Standardlösung pipettieren.
- Die Elektrophoresekammer mit einem Deckel verschließen.

- Die Elektrophoresekammer an ein Spannungsgerät anschließen.
- Am Spannungsgerät eine Spannung von 100 Volt und eine Laufdauer von ca. zwei Stunden einstellen.
- Die Elektrophorese durchführen.
- Nach der anhand des Blaumarkers verfolgten gelelektrophoretischen Auftrennung das Spannungsgerät abschalten und den Gelträger mitsamt dem Gel aus der Kammer nehmen.
- Das vorsichtig vom Gelträger abgelößte Gel auf den UV-Leuchttisch legen.
- Den auf eine Wellenlänge von $\lambda = 312nm$ eingestellten UV-Leuchttisch einschalten.
- Das Gel bei ausgeschalteter sonstiger Dunkelkammerbeleuchtung photographieren.
- Das Gel zurück auf den Gelträger legen und weiterbehandeln oder entsorgen.

Dieses Versuchsprotokoll wurde für *GenLab* überarbeitet und um zusätzliche Informationen zu einem *annotierten Versuchsprotokoll*[Artefakt] ergänzt. Außerdem wurden die einzelnen Arbeitsschritte identifiziert:

A Gelherstellung:

1 Gelmasse herstellen:

a Agarose einwiegen: $1g$ Agarose in dem Messbecher auf der zuvor tarierten Waage abwiegen.

b Puffer hinzugeben: $100ml$ 1x Elektrophoresepuffer (TBE- oder TAE-Puffer) mit dem Messzylinder abmessen und zur Agarose in den Messbecher geben.

Dieser Ansatz dient der Herstellung eines 1%igen Agarosegeles. Höhere Agarosekonzentrationen werden verwandt, um kleinere DNA-Moleküle aufzutrennen.

c Schütteln: Durch Schütteln die Agarose im Puffer suspendieren.

d In Mikrowelle erhitzen: Durch kurzes Aufkochen der Suspension in der Mikrowelle ($ca. 120s$ bei $600W$) die Agarose im Puffer in Lösung bringen.

e EtBr pipettieren: $6\mu l$ Ethidiumbromid-Lösung ($10mg/ml$) zur Agaroselösung hinzupipettieren, so dass eine Ethidiumbromid-Endkonzentration im Gel von $0,6\mu g/ml$ erreicht wird.

Das Ethidiumbromid dient in einem späteren Schritt dem Sichtbarmachen der DNA. Da Ethidiumbromid ein starkes Kanzerogen ist, also krebserregend wirken kann, müssen bei der Arbeit mit dieser Substanz bestimmte Sicherheitsvorkehrungen getroffen werden. So ist insbesondere mit Handschuhen zu arbeiten, und die mit Ethidiumbromid belasteten Abfälle - Pipettenspitzen, Gele etc. - sind gesondert zu sammeln und zu entsorgen.

f Schütteln: Die aufgekochte Lösung durch sachtes Schwenken des Messbechers auf eine einheitliche Konzentration bringen.

2 Geltaschen formen:

a Gelkamm einsetzen: Den Probenkamm in den Gelträger stecken.

b Gel gießen: Die auf etwa $60°C$ abgekühlte Agaroselösung auf den Gelträger gießen.

Die Abkühlzeit beträgt ungefähr 60 *Minuten. Wenn man zu lange wartet, erstarrt die Agaroselösung vorzeitig zu einem Gel und muss erneut aufgekocht werden.*

c Gelkamm entfernen: Nach dem Erstarren der Agaroselösung den Gelkamm vorsichtig aus dem Gel herausziehen.

B Gellauf:

1 Gel vorbereiten:

a TBE-Puffer einfüllen: Die Gelelektrophoresekammer mit TBE-Puffer füllen.

b Gelträger einlegen: Das abgekühlte Gel zusammen mit dem Gelträger aus der Gelkammer herausnehmen und in die Elektrophoresekammer legen.

2 Gel beladen:

a Ansatz pipettieren: Den gesamten Ansatz ($25\mu l$) in eine Geltasche pipettieren.

Dieser Ansatz kann ein Volumen von $5\mu l$ *umfassen, aber auch* $25\mu l$ *oder mehr sind möglich. Falls mehrere Proben analysiert werden sollen, pipettiert man sie in nebeneinander liegende Geltaschen. Man wählt dabei möglichst die Spuren in der Mitte des Geles aus.*

b Standard pipettieren: In eine benachbarte Tasche $15\mu l$ der DNA-Standardlösung pipettieren.

Die DNA-Standards werden in Abhängigkeit von dem interessierenden Größenbereich ausgewählt. So umfasst der DNA-Standard l:EcoRI/HindIII elf Fragmente von $0,56$ *bis* $5,1kb$ *Länge sowie ein weiteres von* $21,2kb$ *Länge. Die Größe von Fragmenten, die in diesem ersten Bereich liegen, können daher recht gut bestimmt werden.*

3 Gel fahren:

a Deckel schließen: Die Elektrophoresekammer mit einem Deckel verschließen.

b Elektrophorese vorbereiten: Die mit einem Deckel verschlossene Elektrophoresekammer an ein Spannungsgerät anschließen, an welchem eine Spannung von 100 Volt und eine Laufdauer von ca. zwei Stunden eingestellt sind.

Achtung - Polung nicht vertauschen, sonst laufen die Proben in die falsche Richtung!

c Gel fahren: Die Elektrophorese durchführen.

d Elektrophorese beenden: Nach der anhand des Blaumarkers verfolgten gelelektrophoretischen Auftrennung das Spannungsgerät abschalten und den Gelträger mitsamt dem Gel aus der Kammer nehmen.

Zur Durchführung der Arbeitsschritte werden eine Reihe von Laborgeräten, Behälter und Substanzen benötigt. Diese werden jeweils in alphabetischer Reihenfolge aufgelistet und die für die Umsetzung des Versuchs benötigten Merkmale werden genannt:

- Laborgeräte:
  - Elektrophoresekammer: Enthält den Gelträger und den Gelkamm.
  - Mikropipetten:
    - ▷ Messbereich: 1 bis $1000\mu l$.
    - ▷ Messgenauigkeit: Abhängig vom Messbereich.
    - ▷ Typen: 20er, 100er, 200er und 1000er (fassen jeweils 20, 100, 200 oder $1000\mu l$).
  - Mikrowelle: Einstellbare Leistung und Zeit.
  - Spannungsgerät: Einstellbare Spannung und Laufdauer.
  - Waagen:
    - ▷ Messbereich: Liegt zwischen $100mg$ und $100g$ (bei der gängigsten Waage).
    - ▷ Messgenauigkeit: Ist unterschiedlich.
- Behälter:
  - Messbecher: Fassen ein Volumen von beispielsweise $50ml$, $600ml$, 1 Liter oder 2 Liter.
  - Messzylinder: Fassen ein Volumen von beispielsweise $50ml$, $100ml$, oder $500ml$ bis hin zu 2 Litern.
- Substanzen:
  - Agarose:
    - ▷ Farbe: Weiß.
    - ▷ Form: Fest.
    - ▷ Schmelzpunkt: 60 bis $95°C$.
  - DNA-Ansatz (Substanzgemisch):
    - ▷ Farbe: Farblos.
    - ▷ Form: Flüssig.
  - DNA-Standardlösung (Substanzgemisch):
    - ▷ Farbe: Farblos.

  - ▷ Form: Flüssig.
- Ethidiumbromid-Lösung ($10mg/ml$):
  - ▷ Farbe: Dunkelrot.
  - ▷ Form: Fest.
  - ▷ Sicherheitshinweise: Krebserregend. Sehr giftig. Reizend.
- Elektrophoresepuffer (Substanzgemisch):
  - ▷ Farbe: Farblos.
  - ▷ Form: Flüssig.
  - ▷ Hinweis: Verhindert das Austrocknen des Gels.
- TBE-Puffer:
  - ▷ Farbe: Farblos.
  - ▷ Form: Flüssig.

Bei der Durchführung der Gelelektrophorese sind bestimmte Abweichungen vom eigentlichen Versuchsprotokoll erlaubt. Diese zeigen sich in der parallelen Ausführbarkeit einzelner Aktivitäten. Außerdem sind eine Reihe von Fehlerfällen möglich. Die Abbildung A.1 zeigt die Gelelektrophorese als Aktivitätsdiagramm mit den Verfeinerungen zum Versuchsprotokoll, den möglichen alternativen Versuchsabläufen (Parallelität) und den Fehlerfällen. Die Fehlerfälle sind jeweils mit einer Nummer und einer entsprechenden Bemerkung versehen. Zu jedem Fehlerfall ist außerdem ein entsprechender Lösungsweg im Aktivitätsdiagramm eingezeichnet.

Des Weiteren werden noch Zustandsdiagramme zur Beschreibung der Zustände der verwendeten Laborgeräte und Behälter erstellt. Damit wird festgelegt, welche Interaktionen mit den Laborgeräten und Behältern möglich sind, das heißt wie sich die Laborgeräte und Behälter in die einzelnen Aktivitäten des Aktivitätsdiagramms integrieren. Ein Beispiel eines Zustandsdiagramms für einen unverschlossenen Behälter ist der Abbildung A.2 zu entnehmen. In Abbildung A.3 ist des Weiteren ein stark vereinfachtes Zustandsdiagramm der Mikrowelle dargestellt. Außerdem ist ein weiteres Beispiel eines Zustandsdiagramms in (Boles u. a., 1998, Seite 45) für die Mikropipette enthalten.

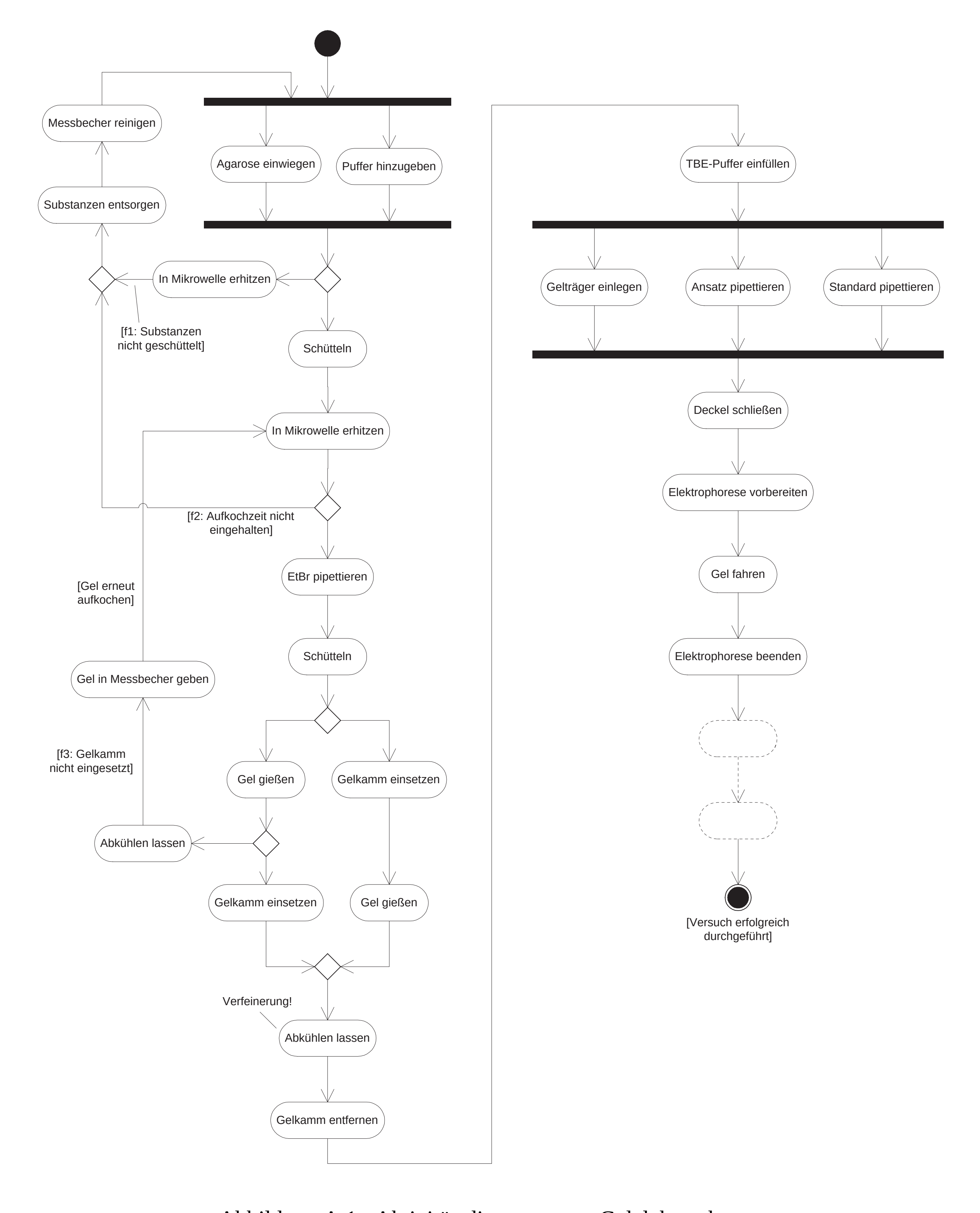


Abbildung A.1.: Aktivitätsdiagramm zur Gelelektrophorese

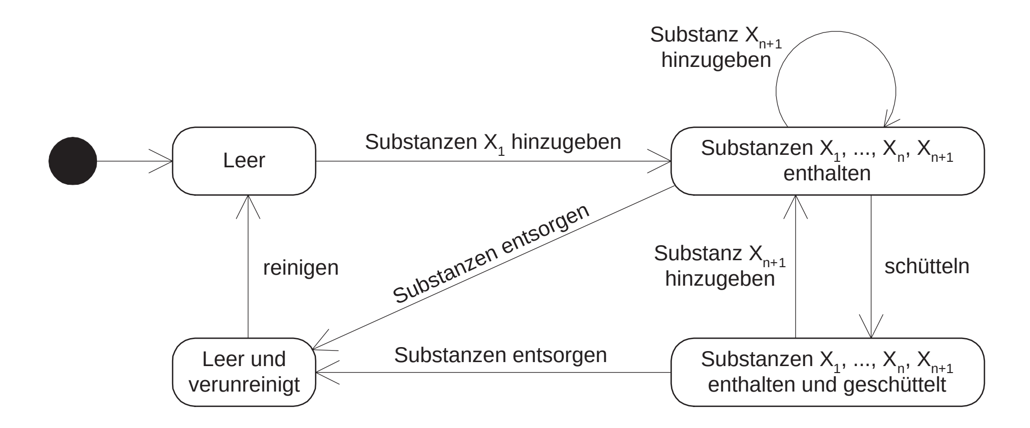


Abbildung A.2.: Zustandsdiagramm eines Behälters (offen)

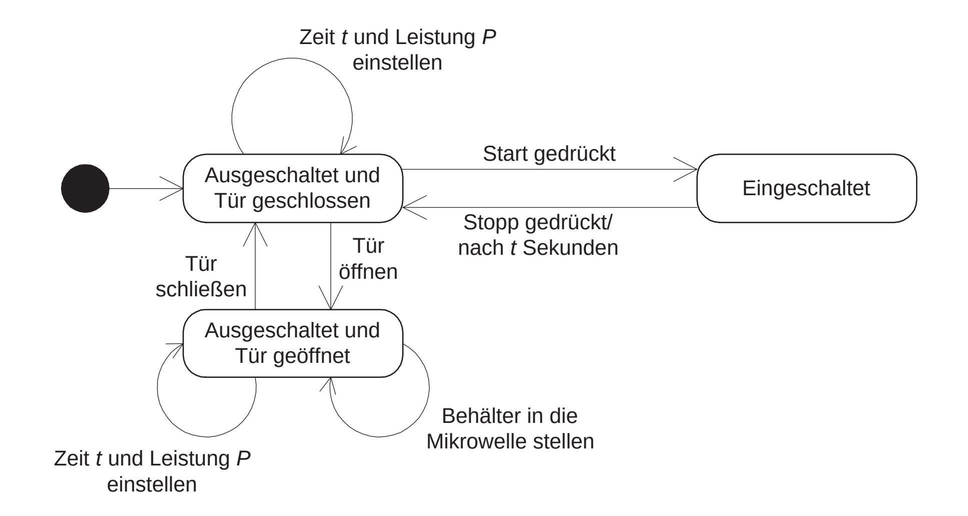


Abbildung A.3.: Vereinfachtes Zustandsdiagramm einer Mikrowelle

B. Checklisten

Die Checklisten dienen vornehmlich zur Unterstützung der Zusammenarbeit des heterogenen Entwicklerteams virtueller Labore. In diesem Kapitel werden die wichtigsten Artefakte aus der Sicht der beteiligten Rollen beschreiben. Dazu wird zu jeder Rolle aufgelistet, was zur Erstellung oder Verwendung der Artefakte zu berücksichtigen beziehungsweise welche Aktivitäten durchzuführen sind.

B.1. Erstellen des Pflichtenhefts

Bevor im Detail die einzelnen Aktivitäten der *Projektmitarbeiter*[Rolle] zur Erstellung des *Pflichtenheftes*[Artefakt] beschrieben werden, wird zunächst das allgemeine Gliederungsschema des *Pflichtenheftes*[Artefakt] vorgestellt:

1. **Einleitung:** Es wird die Domäne des virtuellen Labors kurz beschrieben.
2. **Zielbestimmung:** Es wird beschrieben, welche Ziele durch den Einsatz des virtuellen Labors erreicht werden sollen (siehe *Ersinne neues virtuelles Labor*[Aktivität] in Abschnitt 7.1.1).
3. **Zielgruppe:** Es wird die Zielgruppe des virtuellen Labors beschrieben (siehe *Bestimme die Zielgruppe*[Aktivität] in Kapitel 7.2.1.2).
4. **Lerninhalte:** Es werden die durchzuführenden Versuche nach ihrer Priorität aufgelistet. Dabei ist es auch möglich bestimmte Versuche als optional zu kennzeichnen. Des Weiteren werden die (generischen) Teilversuche und Fertigkeiten identifiziert, das heißt es wird die *Liste der Lerneinheiten*[Artefakt] erstellt (siehe *Analysiere die Domäne*[Aktivität] in Kapitel 7.3.2). Änderungsanfällige Versuche werden gekennzeichnet und die Art der Änderungen wird beschrieben. Zu jedem Versuch, Teilversuch und jeder Fertigkeit werden des Weiteren die benötigten Laborgeräte, Behälter und Substanzen aufgelistet.

   4.1. Versuch 1:

   4.1.1. Laborgeräte
   4.1.2. Behälter
   4.1.3. Substanzen

4.2. Versuch 2:

4.2.1. Laborgeräte

4.2.2. Behälter

4.2.3. Substanzen

4.3. ...

Zu jedem Teilversuch und zu jeder Fertigkeit wird außerdem angegeben, in welchen Versuchen und Teilversuchen diese jeweils enthalten sind beziehungsweise zugehören.

4.1. Teilversuch 1:

4.1.1. Laborgeräte

4.1.2. Behälter

4.1.3. Substanzen

4.1.4. Übergeordnete (Teil-)Versuche

4.2. Teilversuch 2:

4.2.1. ...

4.3. Fertigkeit 1:

4.3.1. Laborgeräte

4.3.2. Behälter

4.3.3. Substanzen

4.3.4. Übergeordnete (Teil-)Versuche

4.4. Fertigkeit 2:

4.4.1. ...

Es kann außerdem das verwendete Quellenmaterial angegeben werden (siehe *Wähle die Wissenserhebungsmethoden*^Aktivität in Kapitel 7.3.1).

5. **Medien:** Falls bereits möglich, werden die Anzahl und Art der Medien, wie zum Beispiel 3D-Modelle, Animationen und Videosequenzen, zur Beschreibung und Darstellung des Hintergrundwissens aufgelistet (siehe *Spezifiziere das Hintergrundwissen zur Lerneinheit* in Kapitel 7.3.5).

   5.1. Theorie zu den Versuchen: Benötigt werden beispielsweise eine textuelle Übersicht und Beschreibung des Versuchsablaufs mit ergänzenden Animationen und das Versuchsprotokoll.

   5.2. Laborgeräte: Die benötigten Medien sind zum Beispiel ein digitalisiertes Photo des realen Laborgerätes, ein 3D-Modell zur Ansicht von allen Seiten, eine kurze Beschreibung sowie eine ausführliche Bedienungsanleitung inklusive der Sicherheitsbestimmungen.

5.3. **Behälter:** Benötigt werden beispielsweise ein digitalisiertes Photo des realen Behälters, ein 3D-Modell zur Ansicht von allen Seiten und eine kurze textuelle Beschreibung.

5.4. **Substanzen** Die benötigten Medien für Substanzen sind zum Beispiel die Strukturformel als Graphik, eine textuelle Beschreibung sowie eine Auflistung weiterer Merkmale gemäß dem Materialblatt.

Wie für die Lerneinheiten kann außerdem das Quellenmaterial zur Erstellung der Medien angegeben werden.

6. **Funktionale Anforderungen:** Zum Beispiel:
   - Durchführen der Versuche, Teilversuche und Fertigkeiten.
   - Aufruf von Hintergrundwissen.
   - Laden und Speichern von Versuchen.

7. **Nicht-funktionale Anforderungen:** Zum Beispiel:
   - Berücksichtigung von Vorgaben des *Corporate Design*.
   - Umsetzung eines vorgegebenen *didaktischen Konzeptes*[Artefakt].

8. **Einsatzumgebung:** Siehe *Bestimme die Einsatzumgebung*[Aktivität] in Abschnitt 7.2.1.3.

9. **Abnahmekriterien:** Es werden die Kritieren festgelegt, die zur erfolgreichen Abnahme des virtuellen Labors führen.

10. **Nutzungsrechte:** Es werden die Nutzungsrechte des virtuellen Labors festgelegt. Diese liegen in der Regel ausschließlich beim *Auftraggeber*[Rolle]. Ausnahme wird allerdings das Framework sein, dass für andere virtuelle Labor wiederverwendet und weiterentwickelt werden soll.

11. **Gesamtentwicklungsplan:** Siehe *Schätze Umfang und Risiko des Projekts*[Aktivität] in Kapitel 7.1.2 sowie *Erstelle den Gesamtentwicklungsplan*[Aktivität] in Kapitel 7.1.3.

Um das allgemeine Gliederungsschema des *Pflichtenheftes*[Artefakt] mit den entsprechenden Inhalten zu füllen, sind von den beteiligten *Projektmitarbeitern*[Rolle] die folgenden Aktivitäten durchzuführen:

- Aktivitäten des *Auftraggebers*[Rolle]:
  - Ziele des virtuellen Labors bestimmen und die Grundlegenden Anforderungen an das virtuelle Labor aufstellen.
  - Auswahl der umzusetzenden Versuche und festlegen der Reihenfolge der Umsetzung in Absprache mit dem *Projektleiter*[Rolle].

  - Ist ein spezielles *didaktisches Konzept*[Artefakt] gewünscht, so ist dieses festzulegen.
  - Trifft das *Pflichtenheft*[Artefakt] die gestellten Anforderungen, dann Unterzeichnen des *Pflichtenheftes*[Artefakt]. Vorher ist zu prüfen:
    - ▷ Stimmt die Beschreibung der Ziele des virtuellen Labors?
    - ▷ Ist die angestrebte Zielgruppe korrekt erfasst worden?
    - ▷ Sind alle gewünschten Lerninhalte enthalten?
    - ▷ Werden alle funktionalen und nicht-funktionalen Anforderungen berücksichtigt?
    - ▷ Sind die Abnahmekriterien des virtuellen Labor akzeptabel?
    - ▷ Wurden die Nutzungsrechte hinreichend geklärt?
    - ▷ Ist die Zeit- und Kostenabschätzung des *Gesamtentwicklungsplans*[Artefakt] akzeptabel?

- Aktivitäten des *Fachexperten*[Rolle]:
  - Analyse der Domäne und Auswahl der Versuche.
  - Identifikation von (generischen) Teilversuchen und Fertigkeiten.
  - Festlegen der Reihenfolge der Versuche.
  - Verwenden des Metaobjektmodells für virtuelle Labore:
    - ▷ Für jeden Versuch und jede Fertigkeit die Laborgeräte, Behälter und Substanzen identifizieren.
    - ▷ Identifikation der zugrunde liegenden Theorie zu den Versuchen.
    - ▷ Identifikation der einzuhaltenden Sicherheitsbestimmungen.
  - Erstellen einer kurzen Beschreibung der Domäne.

- Aktivitäten des *System-Analytikers*[Rolle]:
  - Erstellen eines initialen *Gesamtglossars*[Artefakt] und kontinuierliche Pflege dessen.
  - Identifikation der Anwendungsfälle, neben den Versuchen des virtuellen Labors.
  - Identifikation und abschätzen der potentiellen Risiken bei der Entwicklung des virtuellen Labors (unter Verwendung der Fehlermöglichkeits- und Einflussanalyse).
  - Gegebenenfalls erstellen von *Vorstudien*[Artefakt] zu den kritischen Anwendungsfällen.
  - Definition des virtuellen Labors unter Absprache mit dem *Projektleiter*[Rolle] (die wichtigsten Aktivitäten):
    - ▷ Wahl der Darstellungsart.
    - ▷ Entscheidung über eine Client/Server-Verteilung.
    - ▷ Entscheidung über die Einbindung von Mehrsprachigkeit.
    - ▷ Einbindung von Lernerkontrolle.

        - ▷ Dynamisches Laden und Speichern von Versuchen.

- Aktivitäten des *Medienspezialisten*[Rolle]:
    - Identifiziere die benötigten Medien zur Beschreibung und Darstellung des Hintergrundwissens des virtuellen Labors. Diesbezüglich werden betrachtet:
        - ▷ Die Theorie zu den Versuchen.
        - ▷ Die Laborgeräte, Behälter und Substanzen.
    - Angabe der Quellen zur Erstellung der Medien.

- Aktivitäten des *Projektleiters*[Rolle]:
    - Durchführen einer Bedarfsanalyse zum virtuellen Labor.
    - Bestimmen der Zielgruppe und der Einsatzumgebung.
    - Ziele des virtuellen Labors zusammen mit dem *Auftraggeber*[Rolle] unter Einbeziehung des *Fachexperten*[Rolle] und der *System-Analytiker*[Rolle] erarbeiten.
    - Erheben der Anforderungen des *Auftraggebers*[Rolle] an das virtueller Labor, das heißt insbesondere welche Versuche durchgeführt werden sollen (unter Verwendung von Mind-Maps oder der Qualitätsfunktionen-Darstellung).
    - Abgrenzen der Entwicklung gegenüber bestehenden Konkurrenzprodukten.
    - Zusammenstellen eines potentiellen Entwicklerteams.
    - Initiale Version des *Pflichtenheftes*[Artefakt] erstellen.
    - Identifikation und abschätzen der potentiellen Risiken bei der Entwicklung des virtuellen Labors.
    - Zeit- und Kostenabschätzung der Entwicklung (siehe *Schätze Zeit und Kosten der Entwicklung*[Aktivität] in Abschnitt 7.1.2.1).
    - Prüfen der Ressourcen des Entwicklerteams:
        - ▷ Steht ein *Fachexperte*[Rolle] im hinreichenden Maße während der Entwicklung des virtuellen Labors zur Verfügung?
        - ▷ Ist der *Fachdidaktiker*[Rolle] mit der Entwicklung von multimedialen Lehr- und Lernsystemen vertraut?
        - ▷ Haben die *Informatiker*[Rolle] mit den verwendeten Technologien zur Entwicklung des virtuellen Labors hinreichende Erfahrung?
        - ▷ Haben die *Medienspezialisten*[Rolle] Erfahrung in der Erstellung von realitätsgetreuen Abbildern von Gegenständen der realen Welt?
    - Erstellen des Gesamtentwicklungsplans:
        - ▷ Identifikation der Iterationen, das heißt wann bei der Entwicklung des virtuellen Labors welche Versuche und Fertigkeiten spezifiziert und wann diese realisiert werden sollen.

    - ▷ Erstellen und berücksichtigen der Projektorganisation (Welche Mitarbeiter arbeiten Vollzeit, welche nur Teilzeit?).
    - ▷ Erstellen eines Rollenplans des Entwicklerteams.
    - ▷ Identifikation und Zuordnung der durchzuführenden Aktivitäten zu den Rollen (soweit möglich).
  - Erstellen der endgültigen Version des *Pflichtenheftes*[Artefakt].
  - Unterzeichnen des *Pflichtenheftes*[Artefakt].

B.2. Erstellen eines Anwendungsfalls

Die Aktivitäten der einzelnen Rollen zur Erstellung eines Anwendungsfalls werden am Beispiel des wichtigsten Anwendungsfalls virtueller Labore, dem Anwendungsfall zur Auswahl und Durchführung eines Versuchs, aufgelistet. Es wird dabei davon ausgegangen, dass für die Erstellung der Aktivitätsdiagramme zu den Versuchen entsprechende Entwicklungswerkzeuge verfügbar sind.

- Aktivitäten des *Fachexperten*[Rolle]:
  - Erstellen eines Aktivitätsdiagramms für den Versuchsablauf mit Hilfe des Werkzeuges. Dabei sind insbesondere folgende Punkte zu berücksichtigen:
    - ▷ Identifikation von alternativen Versuchsabläufen, das heißt Arbeitsschritte, deren Reihenfolge vertauscht werden kann (Parallelität im Aktivitätsdiagramm).
    - ▷ Identifikation von möglichen Fehlerfällen.
    - ▷ Erarbeiten und einzeichnen von Lösungswegen zu den Fehlerfällen.
  - Weitergabe des Aktivitätsdiagramms an den *Fachdidaktiker*[Rolle].
  - Evaluation des Versuchs (nach Abschluss der Implementierung).
- Aktivitäten des *Fachdidaktikers*[Rolle]:
  - Prüfen des Aktivitätsdiagramms in Hinblick auf bestehende Lücken und mögliche Vereinfachungen.
  - Bei Unklarheiten bei dem *Fachexperten*[Rolle] nachfragen und gegebenenfalls das Aktivitätsdiagramm überarbeiten lassen.
  - Ist das Aktivitätsdiagramm in Ordnung, dann Weitergabe an den *System-Analytiker*[Rolle].
  - Evaluation des Versuchs (nach Abschluss der Implementierung).
- Aktivitäten des *System-Analytikers*[Rolle]:

  - Prüfen des Aktivitätsdiagramms in Hinblick auf die zur Verfügung stehenden Funktionalitäten des virtuellen Labors. Dabei sind mindestens folgende Fragen zu beantworten:
    - ▷ Welche Laborräume und Arbeitsflächen werden benötigt?
    - ▷ Welche Laborgeräte, Behälter und Substanzen müssen entwickelt werden?
    - ▷ Welche Reaktionsabläufe sind noch nicht im Simulationsmodell enthalten?
  - Bei Unklarheiten den *Fachexperten*[Rolle] oder *Fachdidaktiker*[Rolle] fragen.
  - Weitergabe des Aktivitätsdiagramms an die *Simulationsspezialisten*[Rolle] und die *Designer/Entwickler*[Rolle].

- Aktivitäten des *Simulationsspezialisten*[Rolle]:
  - Erstellen eines Zustandsdiagramms für das Simulationsmodell anhand des *annotierten Versuchsprotokolls*[Artefakt].
  - Bei Unklarheiten den *Fachexperten*[Rolle] oder *Fachdidaktiker*[Rolle] fragen.
  - Implementieren des Zustandsdiagramms des Simulationsmodells in den diskreten Simulator, das heißt implementieren aller Substanzen und Reaktionsabläufe.
  - Testen der Reaktionsabläufe anhand des Zustandsdiagramms des Simulationsmodells.

- Aktivitäten des *Designers/Entwicklers*[Rolle]:
  - Erstellen von Zustandsdiagrammen zu den Laborgeräten und Behältern anhand des *annotierten Versuchsprotokolls*[Artefakt].
  - Bei Unklarheiten den *Fachexperten*[Rolle] oder *Fachdidaktiker*[Rolle] fragen.
  - Implementieren des Versuchsablaufs (Aktivitätsdiagramm) und aller benötigten Laborgeräte und Behälter (Zustandsdiagramme).
  - Testen der Laborgeräte und Behälter anhand der entsprechenden Zustandsdiagramme.

- Aktivitäten des *Testers*[Rolle]:
  - Testen der Implementierung des Simulators anhand des Zustandsdiagramms des Simulationsmodells.
  - Testen der Implementierung der Laborgeräte und Behälter anhand der Zustandsdiagramme.
  - Testen der Implementierung des Versuchsablaufs anhand des Aktivitätsdiagramms.

B.3. Vom Versuchsprotokoll zum virtuellen Versuch

- Aktivitäten des *Fachexperten*[Rolle]:
  - Erweitern des Versuchsprotokolls um die fehlenden Informationen, das heißt erstellen eines *annotierten Versuchsprotokolls*[Artefakt] (siehe *Spezifiziere Aufbau und Ablauf des Versuchs bzw. der Fertigkeit*[Aktivität] in Abschnitt 7.3.4.1):
    - ▹ Auflisten aller verwendeten Laborgeräte, Behälter und Hilfsmittel.
    - ▹ Genaue Beschreibung zur Bedienung der Laborgeräte, Behälter und Hilfsmittel.
    - ▹ Angabe erlaubter Abweichungen beziehungsweise Toleranzgrenzen bei den Mengenangaben.
    - ▹ Anstelle ungenauer oder funktionaler Mengenangaben exakte Mengenangaben machen.
    - ▹ Zu den Substanzen angeben, in welchen Behältern sie sich befinden, wie diese zu lagern und welche Sicherheitsbestimmungen einzuhalten sind.
    - ▹ Zeitangaben werden immer exakt unter Nennung der erlaubten Abweichung angegeben.
    - ▹ Angabe aller Formeln und Regeln zu den Reaktionen und Abläufen.
    - ▹ Mögliche Variationen im Versuchsablauf benennen, das heißt identifizieren von Arbeitsschritten die vertauscht werden können.
    - ▹ Identifikation von Teilversuchen (insbesondere von generischen Teilversuchen) und Fertigkeiten.
  - Bei der Spezifikation der Laborräume und Arbeitsflächen, sowie bei der Umsetzung der Versuchsprotokolle in das Computer-System (Virtualisierung) sind weitere Probleme zu berücksichtigen (siehe dazu *Bestimme Anzahl und Ausstattung der Laborräume*[Aktivität] in Abschnitt 7.2.3.2 und *Spezifiziere Aufbau und Ablauf des Versuchs bzw. der Fertigkeit*[Aktivität] in Abschnitt 7.3.4.1):
    - ▹ Wieviele Laborräume und Arbeitsflächen werden benötigt?
    - ▹ Wie werden die Arbeitsflächen in den Laborräumen angeordnet?
    - ▹ Wie werden die Versuchskomponenten auf die Arbeitsflächen verteilt?
    - ▹ Wie groß ist das größte Laborgerät?
    - ▹ Werden Laborgeräte voraussichtlich zu groß für das virtuelle Labor sein?
    - ▹ Wie ist das Größenverhältnis zwischen der kleinsten und der größten Versuchskomponente?
    - ▹ Sind die Laborgeräte vielleicht deshalb so groß, weil sie veraltet sind?
    - ▹ Existieren modernere Laborgeräte, die eventuell sogar kleiner sind?
    - ▹ Welche Versuchskomponenten sind wahrscheinlich für die maßstabsgetreue Darstellung auf dem Bildschirm zu klein und müssen größer dargestellt werden?

        - ▷ Ist eine sehr hohe Anzahl an Proben in dem Versuch zu erstellen, so ist zu prüfen, ob im virtuellen Labor eine Idealisierung der Proben vorgenommen werden darf, damit auch eine geringere Anzahl ausreicht.
        - ▷ Sind weitere Alternativen in der Versuchsdurchführung möglich, die bisher nicht berücksichtigt wurden? So kann zum Beispiel anstelle eines 37°-Raumes auch ein Heizblock und umgekehrt verwendet werden.
        - ▷ Können unterschiedliche Angaben zu den Geräteeinstellungen in verschiedenen Versuchsprotokollen vereinheitlich werden, damit ein auf die Versuche möglichst optimal zugeschnittenes Laborgerät entwickelt werden kann?
    - Erstellen von Versuchsprotokollen für Teilversuche und Fertigkeiten.
    - Übergeben des *annotierten Versuchsprotokolls*[Artefakt] an die *Fachdidaktiker*[Rolle].
    - Nach Fertigstellung des Versuchs, diesen anhand des *annotierten Versuchsprotokolls*[Artefakt] evaluieren.

- Aktivitäten des *Fachdidaktikers*[Rolle]:
    - Prüfen des *annotierten Versuchsprotokolls*[Artefakt] auf eventuelle Lücken zur Verfeinerung und auf Möglichkeiten zur Vereinfachung.
    - Bei Unklarheiten bei dem *Fachexperten*[Rolle] nachfragen und gegebenenfalls das *annotierte Versuchsprotokoll*[Artefakt] von dem *Fachexperten*[Rolle] überarbeiten lassen.
    - Bei erfolgreicher Prüfung das *annotierte Versuchsprotokoll*[Artefakt] an die *System-Analytiker*[Rolle] weiterreichen.
    - Nach Fertigstellung des Versuchs, diesen anhand des *annotierten Versuchsprotokolls*[Artefakt] evaluieren.

- Aktivitäten des *System-Analytikers*[Rolle]:
    - Prüfen, ob das *annotierte Versuchsprotokoll*[Artefakt] als Grundlage für den Entwurf und die Implementierung verwendet werden kann, das heißt ob die Informationen des *annotierten Versuchsprotokolls*[Artefakt] ausreichen.
    - Bei Unklarheiten bei dem *Fachexperten*[Rolle] oder *Fachdidaktiker*[Rolle] nachfragen.
    - Bei erfolgreicher Prüfung das *annotierte Versuchsprotokoll*[Artefakt] an die *Simulationsspezialisten*[Rolle] und die *Designer/Entwickler*[Rolle] weiterreichen.

- Aktivitäten des *Simulationsspezialisten*[Rolle]:
    - Erstellen eines Zustandsdiagramms für das Simulationsmodell anhand des *annotierten Versuchsprotokolls*[Artefakt].
    - Bei Unklarheiten den *Fachexperten*[Rolle] oder *Fachdidaktiker*[Rolle] fragen.
    - Implementieren des Zustandsdiagramms des Simulationsmodells in den diskreten Simulator, das heißt implementieren aller Substanzen und Reaktionsabläufe.

- Testen der Reaktionsabläufe anhand des Zustandsdiagramms des Simulationsmodells.

- Aktivitäten des *Designers/Entwicklers*[Rolle]:
  - Erstellen von Zustandsdiagrammen zu den Laborgeräten und Behältern anhand des *annotierten Versuchsprotokolls*[Artefakt].
  - Bei Unklarheiten den *Fachexperten*[Rolle] oder *Fachdidaktiker*[Rolle] fragen.
  - Implementierung und Test der benötigten Laborgeräte und Behälter unter Berücksichtigung des *annotierten Versuchsprotokolls*[Artefakt].
  - Integration der getesteten Laborgeräte und Behälter in das virtuelle Labor.
  - Integration des *annotierten Versuchsprotokolls*[Artefakt] in das Anleitungsfenster und Fertigstellung des Versuchs.
- Aktivitäten des *Medienspezialisten*[Rolle]:
  - Erstellung der Medien zu dem Versuch, insbesondere der 3D-Modelle zu den Laborgeräten und Behältern.
  - Integration der Medien zu Abbildern der Laborgeräte und Behälter.
- Aktivitäten des *Testers*[Rolle]:
  - Testen der Implementierung des Simulators anhand des Zustandsdiagramms des Simulationsmodells.
  - Testen der Implementierung der Laborgeräte und Behälter anhand der Zustandsdiagramme.
  - Testen der Implementierung des Versuchsablaufs anhand des *annotierten Versuchsprotokolls*[Artefakt].

B.4. Erstellen eines virtuellen Laborgerätes oder Behälters

- Aktivitäten des *Medienspezialisten*[Rolle]:
  - Festlegen der Modellierungsgenauigkeit unter Absprache mit dem *Fachdidaktiker*[Rolle].
    - ▷ Welche Knöpfe, Schalter und Regler sind wichtig und müssen beweglich sein?
    - ▷ Welche Knöpfe, Schalter und Regler sind unwichtig und brauchen nur grob modelliert werden?
    - ▷ Welche Knöpfe, Schalter und Regler können bei der Modellierung vernachlässigt beziehungsweise weggelassen werden?

    - Welche Einstellungen der Knöpfe, Schalter und Regler des Laborgerätes (Geräteeinstellungen) müssen möglich sein?
    - Wie sieht die Farbgebung aus?
  - Identifikation und Auswahl der Quellen zur Modellierung des Laborgerätes oder Behälters.
  - Erstellen des 3D-Modells unter Verwendung der vorhandenen Quellen.
  - Bei Unklarheiten bei dem *Fachdidaktiker*[Rolle] nachfragen.
  - Prüfen des 3D-Modells, ob die Modellierung den Vorgaben entspricht.
  - Bei der 2D-Darstellung des virtuellen Labors erzeugen einzelner Graphiken des 3D-Modells.
  - Zusammensetzen der einzelnen Graphiken des virtuellen Laborgerätes oder Behälters (gegebenenfalls Verwendung eines entsprechenden Werkzeuges).
  - Bewerten des Erscheinungsbildes des virtuellen Laborgerätes oder Behälters.

- Aktivitäten des *Fachdidaktikers*[Rolle]:
  - Ist die Modellierung didaktisch sinnvoll?
    - Wurden alle wichtigen Knöpfe, Schalter und Regler modelliert?
    - Sind die wichtigen Knöpfe, Schalter und Regler beweglich?
    - Wurden alle unwichtigen Knöpfe, Schalter und Regler weggelassen?
    - Können alle benötigten Einstellungen der Knöpfe, Schalter und Regler des Laborgerätes (Geräteeinstellungen) vorgenommen werden?
  - Bei Unklarheiten bei dem *Fachexperten*[Rolle] nachfragen.
  - Evaluation des Laborgerätes oder Behälters (nach dessen Fertigstellung).

- Aktivitäten des *System-Analytikers*[Rolle]:
  - Analyse des Laborgerätes oder Behälters in Hinblick auf die anstehende Implementierung.
  - Bei Unklarheiten bei dem *Fachdidaktiker*[Rolle] nachfragen.

- Aktivitäten des *Designers/Entwicklers*[Rolle]:
  - Erstellen eines Zustandsdiagramms zu dem Laborgerät oder Behälter.
  - Entwurf entsprechender *Model-*, *View-* und *Controller*-Klassen.
  - Festhalten der Entwurfsentscheidungen in ein Klassendiagramm.
  - Implementierung und Test der Klassen.
  - Bei Unklarheiten bei dem *System-Analytiker*[Rolle] nachfragen.
  - Integration der Medien mit den *View*-Klassen (eventuell auch *Medienspezialist*[Rolle]).

- Integration der *Model*- und *Controller*-Klassen mit den *View*-Klassen und den Medien zu einem fertigen virtuellen Laborgerät oder Behälter.

- Aktivitäten des *Testers*^{Rolle}:
  - Testen der Implementierung des Laborgerätes oder Behälters anhand des Zustandsdiagramms.

- Aktivitäten des *Fachexperten*^{Rolle}:
  - Evaluation des Laborgerätes oder Behälters.
  - Verhält dieses sich so wie in der Realität?

C. Pragmatische Sicht auf den VirtLab–Prozess

Voraussetzung der in Abbildung C.1 dargestellten pragmatischen Sicht auf den *VirtLab*–Prozess ist ein bereits existierendes Framework für virtuelle Labore, das genutzt werden kann. Des Weiteren existieren keine Werkzeuge, die den *VirtLab*–Prozess unterstützen, das heißt die Versuche werden wie in *GenLab* mit Hilfe von *annotierten Versuchsprotokollen*^Artefakt umgesetzt. Zu den einzelnen Phasen der pragmatischen Sicht werden außerdem jeweils nur die wichtigsten Aktivitäten des *VirtLab*–Prozesses aufgelistet.

(1) Vorbereitungsphase:

- *Ersinne neues virtuelles Labor*^Aktivität(siehe Kapitel 7.1.1),
- *Analysiere das Problem*^Aktivität (siehe Kapitel 7.2.1) und
- *Schätze Umfang und Risiko des Projekts*^Aktivität (siehe Kapitel 7.1.2).

(2) Definitionsphase:

- *Definiere das virtuelle Labor*^Aktivität (siehe Kapitel 7.2.3),
- *Definiere die Architektur*^Aktivität (siehe Kapitel 7.4.1),
- *Analysiere die Domäne*^Aktivität (siehe Kapitel 7.3.2) und
- *Erstelle das didaktische Konzept*^Aktivität (siehe Kapitel 7.3.3).

(3a) Planung der Iteration: *Erstelle den Entwicklungsplan für die nächste Iteration*^Aktivität (siehe Kapitel 7.1.4). Dazu sind im Detail durchzuführen:

- *Strukturiere die Medienproduktion*^Aktivität (siehe Kapitel 7.5.4),
- *Strukturiere die Implementierung*^Aktivität (siehe Kapitel 7.6.1),
- *Plane die Integration*^Aktivität (siehe Kapitel 7.6.2),
- *Plane die Tests und Evaluation*^Aktivität (siehe Kapitel 7.7.1),
- *Plane den Einsatz*^Aktivität (siehe Kapitel 7.8.1) und
- *Konzipiere die Entwicklungsumgebung für die Iteration*^Aktivität (siehe Kapitel 7.10.2).

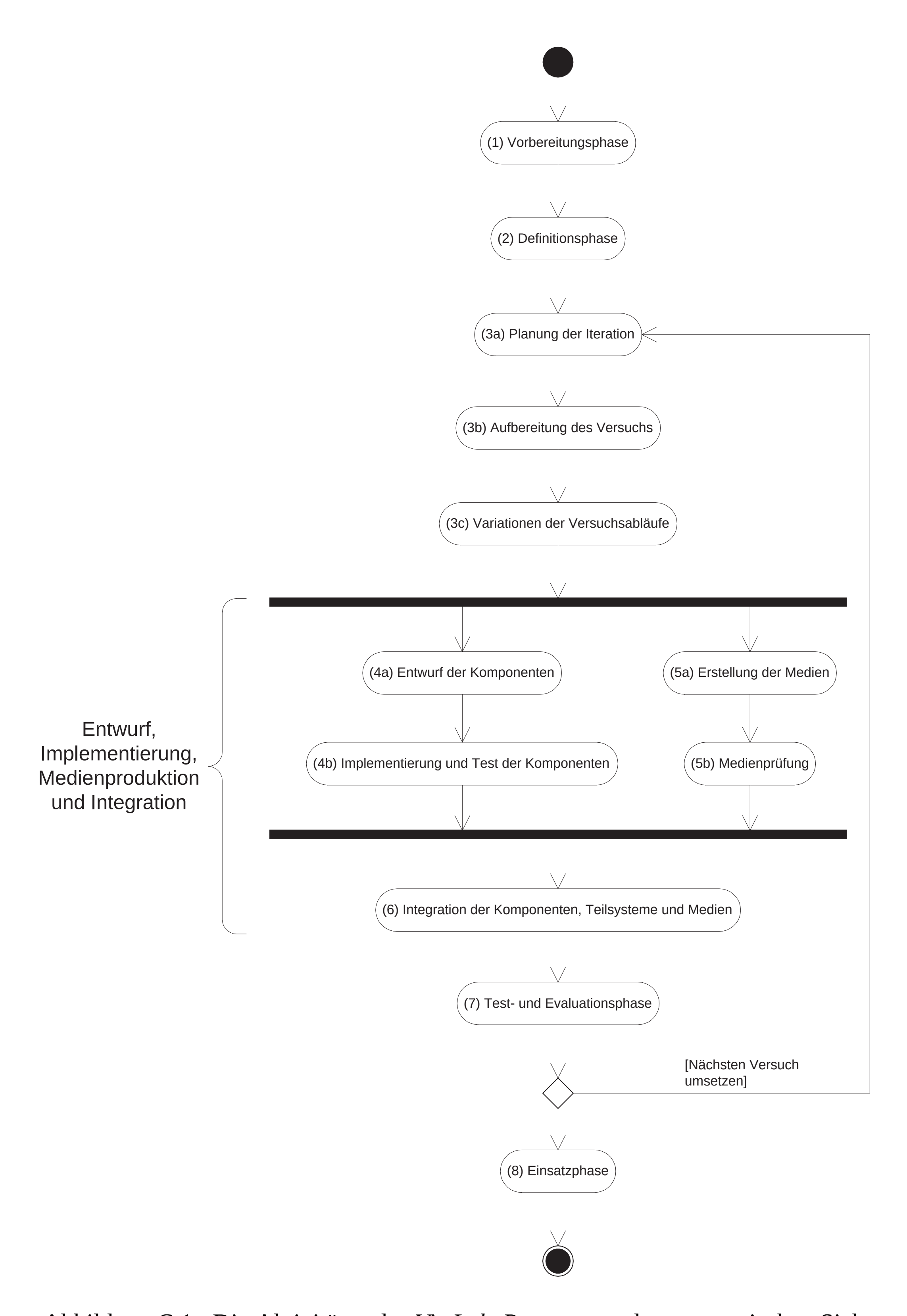


Abbildung C.1.: Die Aktivitäten des *VirtLab*–Prozess aus der pragmatischen Sicht

(3b) Aufbereitung des Versuchs:

- *Spezifiziere Aufbau und Ablauf des Versuchs bzw. der Fertigkeit*[Aktivität] (siehe Abschnitt 7.3.4.1) und
- *Spezifiziere das Hintergrundwissen zur Lerneinheit* (siehe Kapitel 7.3.5).

(3c) Variationen der Versuchsabläufe: *Berücksichtige mögliche Variationen der Versuchsabläufe*[Aktivität] (siehe Abschnitt 7.3.4.2).

(4a) Entwurf der Komponenten: *Entwerfe die Komponenten*[Aktivität] (siehe Kapitel 7.4.4).

(4b) Implementierung und Test der Komponenten: *Implementiere die Komponenten*[Aktivität] (siehe Kapitel 7.6.3).

(5a) Erstellung der Medien:

- *Digitalisiere die Medien*[Aktivität] (siehe Kapitel 7.5.5),
- *Überarbeite die Medien*[Aktivität] (siehe Kapitel 7.5.6),
- *Erstelle die Medien*[Aktivität] (siehe Kapitel 7.5.7) und
- *Beschaffe die Medien*[Aktivität] (siehe Kapitel 7.5.8).

(5b) Medienprüfung: *Prüfe die Medien*[Aktivität] (siehe Kapitel 7.5.9).

(6) Integration der Komponenten, Teilsysteme und Medien:

- *Integriere die Medien*[Aktivität] (siehe Kapitel 7.6.4),
- *Integriere die Komponenten zum Teilsystem*[Aktivität] (siehe Kapitel 7.6.5) und
- *Integriere die Teilsysteme zum Labor*[Aktivität] (siehe Kapitel 7.6.6).

(7) Test- und Evaluationsphase:

- *Führe internen Abnahmetest durch*[Aktivität] (siehe Kapitel 7.8.3),
- *Führe externen Abnahmetest durch*[Aktivität] (siehe Kapitel 7.8.5) und
- *Beende die Iteration*[Aktivität] (siehe Kapitel 7.1.7).

(8) Einsatzphase:

- *Vertreibe das virtuelle Labor*[Aktivität] (siehe Kapitel 7.8.6) und
- *Beende das Projekt*[Aktivität] (siehe Kapitel 7.1.9).

D. Glossar

Adaptierbarkeit: Ein ↑Lehr- und Lernsystem ist dann adaptierbar, wenn es (auf der Grundlage einer extern vorgenommenen Diagnose durch extern vorgenommene Eingriffe) so eingestellt werden kann, dass es dem Unterstützungsbedarf der Lerner möglichst gut entspricht. Siehe dazu im Gegensatz die ↑Adaptivität.

Adaptivität: Ein ↑Lehr- und Lernsystem ist dann adaptiv, wenn das System selbst in der Lage ist, den Unterstützungsbedarf der Lernenden zu diagnostizieren und das Ergebnis der Diagnose in geeignete (angepasste) Lehrtätigkeiten umzusetzen. Siehe dazu im Gegensatz die ↑Adaptierbarkeit.

Akteur (*actor*): Ein Akteur ist eine ↑Rolle, die ein Benutzer des ↑Software-Systems spielt. Akteure befinden sich immer außerhalb des ↑Software-Systems. Akteure können Personen oder externe ↑Software-Systeme sein.

Aktivität (*activity*): Eine Aktivität ist eine Tätigkeit, die bezogen auf ihr ↑Artefakt und ihre Durchführung genau beschrieben werden kann.

Anleitungsfenster: Das Anleitungsfenster enthält eine ↑Kurzanleitung zum ↑Versuchsprotkoll und kann vom ↑Lerner während der Versuchsdurchführung beliebig oft eingesehen werden. Gleichzeitig kann das Anleitungsfenster zur Visualisierung des Lernfortschritts dienen, das heißt zu jedem Arbeitsschritt des ↑Versuchs oder der ↑Fertigkeit wird der aktuelle Stand der Bearbeitung ermittelt und dargestellt.

Anwendungsfall (*use-case*): Ein Anwendungsfall beschreibt die Funktionalität des ↑Software-Systems, die ein ↑Akteur ausführen muss, um ein gewünschtes Ergebnis zu erhalten oder Ziel zu erreichen. Anwendungsfälle sollen den Entwicklern des ↑Software-Systems ermöglichen, mit dem zukünftigen Benutzer über die Funktionalität des ↑Software-Systems zu sprechen, ohne sich gleich in Details zu verlieren und werden im ↑Use-Case-Modell festgehalten. Beispiele für Anwendungsfälle im Kontext virtueller Labore sind die einzelnen ↑Versuche und ↑Fertigkeiten, die durchzuführen beziehungsweise zu erwerben sind, und der Abruf von Hintergrundinformationen zu einem Versuch oder einer Versuchskomponente.

Anwendungsfall-Modell: Das Anwendungsfall-Modell enthält die Menge aller ↑Anwendungsfälle in einem ↑Software-System, das heißt die komplette Funktionalität. Das Anwendungsfall-Modell ersetzt die traditionelle funktionale Beschreibung.

Arbeitsbereich (*workspace*): Der Arbeitsbereich enthält alle Quelltexte, Medien und Dokumente an denen ein Projektmitarbeiter arbeitet. Ebenfalls zum Arbeitsbereich gehören verwendete Programm-Bibliotheken und Gerätetreiber.

Architektur (*architecture*): Die Architektur eines ↑Software-Systems bestimmt dessen Strukturierung in ↑Teilsysteme und ↑Komponenten, und legt fest, welche Beziehungen zwischen diesen Teilsystemen und Komponenten existieren und wie sie miteinander interagieren. Dabei sind nur die *Black-Box*-Eigenschaften der Komponenten von Interesse, das heißt nur diejenigen Informationen, die für die Benutzung der Komponenten benötigt werden. Die Architektur ist zentraler Bestandteil eines virtuellen Labors und wird mit Hilfe eines ↑Frameworks realisiert. Der stabile Kern dieses ↑Frameworks stellt das ↑OOD-Modell dar.

Artefakt (*artifact*): Artefakte sind Ergebnisse, die im Laufe der Software-Entwicklung entstehen. Artefakte werden von Projektmitarbeitern erzeugt, geändert und benutzt, wenn diese ↑Aktivitäten ausführen.

Autorensystem: Autorensysteme sind graphisch-interaktive ↑Software-Systeme, die den technischen Entwicklungsprozess multimedialer Anwendungen (↑Multimedia-Systeme) mit Hilfe von Konzepten der ↑visuellen Programmierung unterstützen. Sie ermöglichen eine graphisch-interaktive Spezifikation der Beziehungen zwischen Medienobjekten, die idealerweise durch eine Programmiersprache (in der Regel eine Skriptsprache) unterstützt wird.

Baseline: Eine Baseline ist eine geprüfte und gebilligte Freigabe von ↑Artefakten, die eine gemeinsame Basis für die weitere Evolution oder Entwicklung eines ↑Software-Systems darstellt und die nur durch einen *formalen Antrag* (zum Beispiel im Konfigurationsmanagement) geändert werden kann.

Build: Ein Build ist eine ablauffähige Version eines (Teil-)Systems, das eine Teilmenge der Funktionalitäten des endgültigen Produktes demonstriert.

CASE: Computerunterstützte Software-Entwicklung (*Computer Aided Software Engineering*), das heißt der Einsatz von ↑CASE-Werkzeugen bei der Software-Entwicklung.

CASE-Plattform: Software-Produkt, das allgemeine Basisdienstleistungen wie Benutzungsschnittstelle und Datenhaltung zur Verfügung stellt und damit ↑CASE-Werkzeuge von diesen Dienstleistungen entlastet. In eine CASE-Plattform können ↑CASE-Werkzeuge integriert werden (↑CASE-Umgebung).

CASE-Umgebung: Software-Entwicklungsumgebung, die konzeptionell aus einer ↑CASE-Plattform und mehreren darin integrierten ↑CASE-Werkzeugen besteht.

CASE-Werkzeug: In der ↑Software-Technik versteht man unter einem CASE-Werkzeug ein Software-Produkt, das als Hilfsmittel zur Entwicklung von ↑Software eingesetzt wird. Allgemein dienen CASE-Werkzeuge zur automatischen Unterstützung von Methoden, Verfahren und Notationen.

CBT (*Computer Based Training*): Eine Bezeichnung für computerunterstützte ↑Lehr- und Lernsysteme, die didaktisch aus den Ansätzen des Instruktionsdesigns, das heißt der Unterrichtsplanung beziehungsweise des Unterrichtsentwurfs der siebziger Jahre abgeleitet werden. Computer Based Training ist für bestimmte Lernformen wie zum Beispiel Informationsvermittlung und *Drill & Practice* gut einsetzbar.

Client/Server-Architektur: Verteilung einer — in mehrere logische Software-Schichten (↑Schichtenarchitektur) gegliederten — Anwendung auf ein Netzwerk, das aus vielen *Clients* mit mindestens einem *Server* besteht.

Didaktik: Teilbereich der Pädagogik, der die Wissenschaft vom Lehren und Lernen umfasst. Die Didaktik befasst sich dabei vornehmlich mit der Erforschung von Lehr- und Lern-Prozessen im Allgemeinen, mit deren vorlaufenden Bedingungen, deren Initiierung und Förderung sowie mit deren Ergebnissen.

Director: Kommerzielles ↑Autorensystem der Firma *Macromedia*.

Domäne: Eine Domäne ist ein Diskursbereich (Ausschnitt) der Realität, also ein bestimmter Arbeits- oder Wissensbereich beziehungsweise ein Spezialgebiet.

Dynamischer Test: Dynamische Testverfahren führen ein ausführbares zu testendes ↑Software-System oder ein ausführbares zu testendes ↑Teilsystem auf einem Computer-System aus. Siehe dazu im Gegensatz die ↑Statische Analyse.

Echtzeit: Wird ein Modell, wie zum Beispiel ein 3D-Modell oder ein Simulationsmodell, durch die unmittelbare Reaktionszeit des Computer-Systems auf die Manipulation des Benutzers hin interaktiv veränderbar, so wird diese Reaktionszeit als Echtzeit bezeichnet. Dabei ist die Reaktionszeit des Computer-Systems abhängig von dessen Leistungsfähigkeit, der zu bewegenden Datenmenge und der Bewegungsgeschwindigkeit durch den Benutzer.

Entwicklungsmethodik: Siehe Kapitel 2.2.

Entwicklungsplattform: Siehe ↑CASE-Plattform.

Entwicklungsumgebung: Siehe ↑CASE-Umgebung.

Entwicklungswerkzeug: Siehe ↑CASE-Werkzeug.

Ergebnis: Siehe ↑Artefakt.

Experiment: Siehe ↑Versuch.

Expertensystem: Ein Expertensystem ist ein ↑Software-System, das Spezialwissen und Schlussfolgerungsfähigkeit von Experten auf einem eng begrenzten Aufgabengebiet (siehe ↑Domäne) rekonstruiert.

Externes Release: Siehe ↑Release.

Fertigkeit (*skill*): Eine Fertigkeit ist eine bestimmte, in sich abgeschlossene Handlungsabfolge (↑Teilversuch), die in einem oder mehreren ↑Versuchen durchzuführen ist. Ziel einer Fertigkeit ist es, dass die Handlungsabfolge vom Lerner verarbeitet und im Gedächnis gespeichert wird, um sie bei entsprechenden ↑Versuchen wieder abrufen und anwenden zu können. Analog zu den ↑Versuchen und ↑Teilversuchen können auch für die Fertigkeiten entsprechende ↑Versuchsprotokolle erstellt werden.

Framework: Ein Framework besteht aus einer Menge von zusammenarbeitenden Klassen, die einen wiederverwendbaren Entwurf für einen bestimmten Anwendungsbereich implementieren. Es besteht aus konkreten und insbesondere abstrakten Klassen. Im Allgemeinen wird vom Anwender (also dem Programmierer) des Frameworks erwartet, dass er Unterklassen definiert, um das Framework zu verwenden und anzupassen. Siehe dazu ↑Architektur.

Generischer Teilversuch: Ein generischer Teilversuch ist ein ↑Teilversuch, der erst noch durch die Angabe konkreter Substanzen instantiiert werden muss. Die durch einen generischen Teilversuch beschriebene Basistechnik ist prinzipiell auf verschiedene Substanzen anwendbar.

GenLab: Am Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme (OFFIS) in der Arbeitsgruppe ↑Multimedia-Systeme von Prof. Dr. H.-J. Appelrath seit September 1997 entwickeltes multimediales gentechnisches Praktikum (*GenLab*). Mit *GenLab* können gentechnische ↑Versuche in einer virtuellen Umgebung auf einem handelsüblichen Computer-System durchgeführt werden. Siehe dazu ↑*VirtLab*.

Geschäftsprozess (*workflow, business use case*): Ein Geschäftsprozess besteht aus mehreren zusammenhängenden Aufgaben, die durchgeführt werden, um ein Ziel zu erreichen beziehungsweise ein gewünschtes Ergebnis zu erstellen.

Intelligentes Tutorsystem: Siehe Kapitel 2.7.

Interaktivität: Umfassender Begriff für solche Eigenschaften eines Computer-Systems, die dem Benutzer Eingriffs- und Steuermöglichkeiten eröffnen, im Idealfall auch die wechselnde Dialog-Initiative von Mensch und Computer.

Internes Release: Siehe ↑Release.

Iteration: Iterationen zerlegen den Software-Entwicklungsprozess beziehungsweise die ↑Phasen des Software-Entwicklungsprozesses in mehrere gleichartige Schritte.

Iterationsmeilenstein: Ein ↑Meilenstein, der das Ende einer ↑Iteration definiert. Iterationsmeilensteine unterteilen ↑Phasen in mehrere Schritte und legen wichtige Zwischenergebnisse (↑Artefakte) fest. Die Festlegung der Iterationsmeilensteine ist nicht so sehr an die inhaltliche Gliederung der Software-Entwicklung gebunden, sondern verfolgt die Erreichung von zuvor vereinbarten Zielen.

Java: Plattformunabhängige, objektorientierte Programmiersprache entwickelt von der Firma *Sun Microsystems*.

Java-Applet: Ein *Java-Applet* ist ein Software-System, das in der Programmiersprache ↑Java entwickelt wurde und in Verbindung mit einem Java-fähigen Web-Browser eingesetzt werden kann.

Klasse (*class*): Eine Klasse definiert für eine Kollektion von Objekten deren *Struktur* (Attribute), *Verhalten* (Operationen) und *Beziehungen* (Assoziationen und Vererbungsstrukturen). Klassen besitzen einen Mechanismus, um neue Objekte zu erzeugen.

Komponente (*component*): Eine Komponente ist ein nicht-triviales, nahezu unabhängiges und ersetzbares Teil eines ↑Software-Systems, dass eine eindeutige Funktion im Kontext einer wohldefinierten ↑Software-Architektur erfüllt und nach außen über Schnittstellen zur Verfügung gestellt wird. Beispiele für Komponenten im Kontext virtueller Labore sind die Arbeitsflächen, Versuchskomponenten und deren gegenseitiges Zusammenwirken im Simulationsmodell. Weitere Beispiele sind das Lerner-, Lehrer- und Expertenmodul des ↑intelligenten Tutorsystems, sowie die Transportleiste und das Anleitungsfenster.

Kurzanleitung: Die Kurzanleitung ist eine sehr knapp gehaltene Übersicht über einen kompletten ↑Versuch, das heißt eine sehr stark gekürzte Version des entsprechenden ↑Versuchsprotokolls.

Laborübersicht: Eine Laborübersicht ermöglicht es, zwischen den Arbeitsflächen des virtuellen Labors zu wechseln und wird beispielsweise mit einer Grundrissansicht auf das Labor oder einem Auswahlmenü realisiert.

Lehr- und Lernsystem: Siehe Kapitel 2.4.

Lerner: Ein Individuum beschäftigt mit dem Erwerb von Wissen oder Fertigkeiten mit Hilfe eines ↑Lehr- und Lernsystem oder ↑intelligenten Tutorsystems.

Lerneinheit: Eine Lerneinheit besteht aus einer Menge von zu vermittelndem Wissen (*Lerninhalt*) und dem Kenntnisstand des ↑Lerners nach Erwerb des Wissens (*Lernziel*). Im Kontext virtueller Labore ist eine Lerneinheit ein ↑Versuch oder ↑Teilversuch der durchgeführt oder eine ↑Fertigkeit die erworben werden soll.

Lerneranalyse: Unter der Lerneranalyse wird die ständige Beobachtung, das heißt Protokollierung und Analyse der Aktivitäten des ↑Lerners mit dem Zweck der Erstellung des Lernermodells und der Messung und Bewertung des Lernfortschritts (Effektivität und Effizienz des Lernens) verstanden.

Lerninhalt: Im Kontext virtueller Labore sind die Lerninhalte die vom ↑Lerner im virtuellen Labor durchzuführenden ↑Versuche und deren zugrunde liegendes theoretisches Hintergrundwissen. Siehe auch ↑Lerneinheit.

Lernziel: Siehe ↑Lerneinheit.

Medium: Mittel zur Verbreitung und Darstellung von Information. Ein Medium heißt *diskret* (oder *zeitunabhängig* oder *statisch*), wenn seine Verarbeitung zeitunkritisch ist, wie zum Beispiel Text, Graphiken und Diagramme. Ein Medium heißt *kontinuierlich* (oder *zeitabhängig* oder *dynamisch*), wenn seine darzustellenden Informationen nicht nur durch ihre Werte vermittelt werden, sondern auch durch den zeitlichen Verlauf ihres Auftretens, wie zum Beispiel Audiosequenz, Videosequenz und Animation.

Meilenstein: Ein Meilenstein definiert einen festgelegten Zeitpunkt (*Termin*), zu dem eine Menge von ↑Artefakten in vorgegebener Detaillierung und Vollständigkeit nachprüfbar und formal dokumentiert vorliegen soll. Ein Meilenstein stellt ein Hilfsmittel zur Planung und Überwachung eines Projektes dar. Es wird des Weiteren unterschieden in ↑Iterationsmeilenstein und ↑Phasenmeilenstein.

Multimedia-System: Siehe Kapitel 2.3.

Multimediales Lehr- und Lernsystem: Siehe Kapitel 2.5.

OOA-Modell: Das OOA-Modell ist die fachliche Lösung des zu realisierenden ↑Software-Systems, die in einer objektorientierten Notation modelliert wird. Das OOA-Modell ist das wichtigste Ergebnis der Analyse.

OOD-Modell (*design model*): Das OOD-Modell ist die technische Lösung des zu realisierenden ↑Software-Systems, die in einer objektorientierten Notation modelliert wird. Das OOD-Modell ist ein Abbild des späteren objektorientierten Programms.

Phase: Zusammenfassung von ↑Aktivitäten der Software-Entwicklung nach zeitlichen, begrifflichen und organisatorischen Kriterien.

Phasenmeilenstein: Ein ↑Meilenstein, der das Ende einer ↑Phase definiert. Phasenmeilensteine gliedern die Software-Entwicklung in inhaltliche Einheiten und sind eng verknüpft mit den durchzuführenden ↑Aktivitäten.

Pilotsystem: Ein Pilotsystem ist ein ↑Software-System, das eine komplette Anwendung realisiert, also einschließlich Benutzungsoberfläche und Datenhaltung. Es bietet jedoch nicht die volle Funktionalität, sondern nur die elementaren Funktionen an. Im Gegensatz zum ↑Prototypen kann das Pilotsystem zu einem einsatzfähigen Software-Produkt weiterentwickelt werden.

Prototyp: Ein Prototyp ist ein ↑Software-System, das bestimmte Aspekte eines — später zu entwickelnden Systems — vorweg nimmt. Häufig werden Prototypen der Benutzungsoberfläche (Oberflächenprototyp) eingesetzt oder dienen beispielsweise zur Evaluierung der technischen Datenbankanbindung. Prototypen werden im Gegensatz zu ↑Pilotsystemen im Allgemeinen „weggeworfen“.

Release: Das Ergebnis einer jeden ↑Iteration ist ein neues Release. Ein Release kann *intern* oder *extern* sein. Ein internes Release (α*-Release*) wird nur als Meilenstein oder zur Demonstration, Test und Evaluation beim Auftraggeber oder zukünftigen Endanwender verwendet. Ein externes Release (β*-Release*) wird dagegen an den Auftraggeber oder Endanwender ausgeliefert beziehungsweise verkauft.

Release-Zyklus: Ein Release-Zyklus ist die Dauer zwischen zwei externen ↑Releases.

Repository (*repository***):** Speicherbereich für die Quelltexte, Medien und Dokumente des virtuellen Labors.

Review: Ein Review ist eine Sitzung zur kritischen Begutachtung des aktuellen Projektstands auf Basis repräsentativer Zwischenergebnisse (↑Artefakte) mit dem Ziel, den Status zu ermitteln und mögliche Vorschläge zur weiteren Problemlösung aufzunehmen.

Rolle (*role***):** Eine Rolle beschreibt die notwendigen Erfahrungen, Kenntnisse und Fähigkeiten, über die ein Projektmitarbeiter verfügen muss, um eine bestimmte ↑Aktivität durchzuführen.

Schichtenarchitektur: Gliederung der ↑Software-Architektur in hierarchische Schichten. Zwischen den Schichten kann eine lineare, strikte oder baumartige Ordnung bestehen. Anwendungen werden oft nach einer Drei–Schichten–Architektur aufgebaut: Benutzungsoberfläche, eigentliche Anwendung und Datenhaltung.

Schnittstelle (*interface***):** Eine Schnittstelle ist eine Sammlung von Operationen um die Dienste einer ↑Klasse oder ↑Komponente zu spezifizieren.

Shockwave-Film: Ein *Shockwave-Film* ist ein mit Hilfe des ↑Autorensystems ↑*Director* erstelltes ↑Software-System, das wie ein ↑*Java-Applet* mit einem dafür geeigneten Web-Browser ausgeführt werden kann.

Simulator: Siehe Kapitel 2.6.

Software: Unter Software versteht man die Gesamtheit aller Programme, die auf einem Computer-System eingesetzt werden können.

Software-Architektur: Siehe ↑Architektur.

Software-System: Ein aus ↑Software bestehendes System.

Software-Qualität: Unter Software-Qualität versteht man nach DIN ISO 9126 die Gesamtheit der Merkmale und Merkmalswerte eines ↑Software-Systems, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen. Des Weiteren fließen für virtuelle Labore auch inhaltliche, didaktische und gestalterische Qualitätsmerkmale mit ein.

Software-Technik: Anwendung von Prinzipien, Methoden und Techniken auf den Entwurf und die Implementierung von ↑Software-Systemen. Der Begriff Software-Technik steht für die Auffassung, dass die Erstellung, Anpassung und Wartung von ↑Software-Systemen kein künstlerischer, sondern vorwiegend ein ingenieurmäßig ablaufender Prozess ist.

Statische Analyse: Überprüfung, Vermessung und Darstellung eines ↑Software-Systems oder ↑Teilsystems durch Analyse des Quelltextes unter Verzicht auf die Durchführung von ↑dynamischen Tests.

Teilsystem (*subsystem*): Ein Teilsystem ist ein eigenständiger Teil eines ↑Software-Systems. Das Verhalten eines Teilsystems wird bestimmt durch die ↑Klassen und weiteren Teilsysteme, die es enthält. Ein Teilsystem hat dabei eine oder mehrere ↑Schnittstellen, über die auf das Teilsystem zugegriffen werden kann und die sein Verhalten definieren. Beispiele für Teilsysteme im Kontext virtueller Labore sind der ↑Simulator und das ↑intelligente Tutorsystem.

Teilversuch: Ein Teilversuch stellt in der Regel eine bestimmte Basistechnik der zu modellierenden ↑Domäne dar, die zur Erreichung wichtiger Zwischenschritte und Teilergebnisse im ↑Versuchsablauf dienen. Ein Teilversuch kann in verschiedenen ↑Versuchen auftauchen oder ist innerhalb eines ↑Versuches mehrfach durchzuführen. Des Weiteren kann ein Teilversuch auch als eigenständiger ↑Versuch aufgefasst werden und ebenfalls aus weiteren Teilversuchen bestehen. Siehe dazu auch ↑generischer Teilversuch und ↑Fertigkeit.

Tooltip: Unter einem Tooltip wird eine kurze Beschreibung eines Knopfes oder Symbols auf der Benutzungsoberfläche bezeichnet. Ein Tooltip erscheint auf der Benutzungsoberfläche, wenn der Mauszeiger für kurze Zeit über dem Oberflächenelement stehen bleibt.

Transportleiste: Das Konzept der Transportleiste realisiert im virtuellen Labor die Möglichkeit Versuchskomponenten, das heißt Laborgeräte, Behälter und Substanzen, zwischen verschiedenen Arbeitsflächen zu transportieren.

Unified Modeling Language (UML): Notationssprache für die objektorientierte Software-Entwicklung. Die Unified Modeling Language wurde von Booch, Rumbaugh und Jacobsen bei der *Rational Software Corporation* entwickelt und 1997 von der *Object Management Group* (OMG) als Standard akzeptiert.

Use-Case: Siehe ↑Anwendungsfall.

Use-Case-Modell (*use-case model*): Siehe ↑Anwendungsfall-Modell.

Versuch (*experiment*): Ein Versuch bezeichnet im (virtuellen) Labor die Abfolge von bestimmten Arbeitsschritten nach einem entsprechenden ↑Versuchsprotokoll. Ein Versuch kann aus mehreren ↑Teilversuchen bestehen.

Versuchsablauf: Ein Versuchsablauf bezeichnet die Durchführung eines ↑Versuchs in einer bestimmten Abfolge von Arbeitsschritten.

Versuchsprotokoll: Ein Versuchsprotokoll enthält eine knappe und in der Regel sequenzielle Beschreibung der Arbeitsschritte, die für die Durchführung eines ↑Versuchs notwendig sind.

VirtLab: Auf der Basis der Ergebnisse und Erfahrungen von ↑*GenLab* werden in *VirtLab* seit September 2000 Methoden und Werkzeuge, also Vorgehensmodelle, Notationen, Entwurfsmuster, Frameworks und Spezifikationssprachen, zur Entwicklung allgemeiner multimedialer virtueller naturwissenschaftlich-technischer Labore und Praktika erarbeitet. Erklärtes Ziel von *VirtLab* ist es, die schnelle und kostengünstige Produktion qualitativ hochwertiger virtueller Labore zu ermöglichen.

Virtuelle Realität: ↑Virtuelle Realität ist eine durch ein Computer-System erzeugte Modellwelt, in der im ↑Echtzeitverhalten navigiert werden kann, mit dem Anspruch, dem Benutzer durch koordinierte Stimulierung von Sinnesorganen mittels geeigneter Geräte den Eindruck zu vermitteln, sich in dieser Modellwelt zu befinden.

Virtuelles Labors: Siehe Kapitel 2.8.

Vorgehensmodell: Siehe Kapitel 2.1.

Werkzeug: Siehe ↑CASE-Werkzeug.

Workflow: Siehe ↑Geschäftsprozess.

Zwischenergebnis: Siehe ↑Artefakt.

E. Literaturverzeichnis

[Aden u. a. 1998] ADEN, Thomas ; APPEL, Jörg ; BOLES, Dietrich ; DAWABI, Peter ; HEUTEN, Wilko ; KOOPMANN, Stefan ; KRÜGER, Tobias ; LINDNER, Marco ; SACHTELEBEN, Mario ; SCHLATTMANN, Marco ; SCHNEIDER, Holger ; WICHMANN, Christian: *Zwischenbericht Teil B der Projektgruppe Virtuelle naturwissenschaftlich-technische Labore im Internet*. Oldenburg : Carl von Ossietzky Universität Oldenburg - Fachbereich Informatik, September 1998. – URL `http://elvis.offis.uni-oldenburg.de/pdf/Zwischenbericht-B.pdf`

[Aden u. a. 1999] ADEN, Thomas ; APPEL, Jörg ; BOLES, Dietrich ; DAWABI, Peter ; HEUTEN, Wilko ; KOOPMANN, Stefan ; KRÜGER, Tobias ; LINDNER, Marco ; SACHTELEBEN, Mario ; SCHLATTMANN, Marco ; SCHNEIDER, Holger ; WICHMANN, Christian: *Endbericht der Projektgruppe Virtuelle naturwissenschaftlich-technische Labore im Internet*. Oldenburg : Carl von Ossietzky Universität Oldenburg - Fachbereich Informatik, März 1999. – URL `http://elvis.offis.uni-oldenburg.de/pdf/Endbericht.pdf`

[Alsdorf und Bannwart 1997] ALSDORF, Claudia ; BANNWART, Edouard: Virtuelle Realität: Erfahrbare Informationen im Cyberspace. In: ISSING, Ludwig J. (Hrsg.) ; KLIMSA, Paul (Hrsg.): *Information und Lernen mit Multimedia*. 2., überarbeitete Auflage. Weinheim : Psychologie Verlags Union, Januar 1997, S. 436–450

[Ambler 2001] AMBLER, Scott W.: *Enterprise Unified Process: Enhancing the Unified Process to Meet the Real-World Needs of Your Organization*. Ronin International, April 2001. – URL `http://www.ronin-intl.com/publications/unifiedProcess.PDF`

[ANSSTAND e.V. 2001] ANSSTAND E.V.: *Anwender des Software-Entwicklungsstandards der öffentlichen Verwaltung (ANSSTAND) e.V. - Homepage*. Deisenhofen : ANSSTAND e.V., 2001. – URL `http://www.ansstand.de/`

[Appelrath u. a. 1998] APPELRATH, Hans-Jürgen ; BOLES, Dietrich ; CLAUS, Volker ; WEGENER, Ingo: *Starthilfe Informatik*. Stuttgart, Leipzig : B. G. Teubner, 1998

[Appelrath und Ludewig 2000] APPELRATH, Hans-Jürgen ; LUDEWIG, Jochen: *Skriptum Informatik: eine konventionelle Einführung*. 5. Auflage. Stuttgart : B. G. Teubner, 2000

[Appelrath und Schlattmann 1999] APPELRATH, Hans-Jürgen ; SCHLATTMANN, Marco: Virtuelle multimediale Labore und Praktika. In: *OFFIS Jahresbericht 1999* (1999). – URL `http://www.offis.de/img/jahresbericht/jb99/pdfs/mi-01.pdf`

[Aquilano u. a. 1998] AQUILANO, Nicholas J. ; CHASE, Richard B. ; JACOBS, F. Robert: *Production and operations management: manufacturing and services*. 8th edition. McGraw Hill Companies, Inc., 1998

[Ateyeh u. a. 1999] ATEYEH, Khaldoun ; MÜLLE, Jutta A. ; LOCKEMANN, Peter C.: *Modulare Aufbereitung von multimedialen Lerninhalten für eine heterogene Lernumgebung*. 1999. – URL `http://vikar.ira.uka.de/publikationen/`

[Balzert 2000a] BALZERT, Heide: *Objektorientierung in 7 Tagen : vom UML-Modell zur fertigen Web-Anwendung*. Heidelberg, Berlin : Spektrum, Akad. Verl., 2000 (Lehrbücher der Informatik). – Fachhochschule Dortmund - Fachbereich Informatik

[Balzert u. a. 1993] BALZERT, Heide ; BALZERT, Helmut ; LIGGESMEYER, Peter ; BALZERT, Helmut (Hrsg.): *Systematisches Testen mit Tensor*. Mannheim, Leipzig, Wien, Zürich : BI-Wissenschaftsverlag, 1993 (Angewandte Informatik Band 9)

[Balzert 2000b] BALZERT, Helmut: *Lehrbuch der Software-Technik*. Bd. 1. Software-Entwicklung. 2. Auflage. Heidelberg, Berlin : Spektrum, Akad. Verl., 2000. – Ruhr-Universität Bochum

[Barchfeld u. a. 2000] BARCHFELD, Markus ; SAND, Roland ; LINK, Johannes: XP und RUP - Passt das zusammen? In: *Net Object Days 2000* (2000)

[Baumgartner 1997] BAUMGARTNER, Peter: Didaktische Anforderungen an (multimediale) Lernsoftware. In: ISSING, Ludwig J. (Hrsg.) ; KLIMSA, Paul (Hrsg.): *Information und Lernen mit Multimedia*. 2., überarbeitete Auflage. Weinheim : Psychologie Verlags Union, Januar 1997, S. 241–251

[Beck 2000] BECK, Kent: *Extreme programming explained: embrace design*. Addison-Wesley, Dezember 2000

[Blumstengel 1998] BLUMSTENGEL, Astrid: *Entwicklung hypermedialer Lernsysteme*. Berlin : Wissenschaftlicher Verlag Berlin, 1998. – Dissertation

[BmBF 2001] BMBF: *L3: Lebenslanges Lernen - Weiterbildung als Grundbedürfnis*. 2001. – URL `http://www.l-3.de/`

[Boedicker 1990] BOEDICKER, Dagmar: *Handbuch-Knigge: Software-Handbücher schreiben und beurteilen*. Mannheim, Wien, Zürich : BI-Wissenschaftsverlag, 1990 (Angewandte Informatik Band 6)

[Boles 1994] BOLES, Dietrich: *Das IMRA-Modell - Integrationen von Interaktionen in das Autorenwerkzeug FMAD*. Oldenburg : Carl von Ossietzky Universität Oldenburg — Fachbereich Informatik, September 1994. – Diplomarbeit

[Boles 1997] BOLES, Dietrich: Erstellung multimedialer Dokumente und Anwendungen. In: *Workshop Software-Engineering für Multimedia-Systeme im Rahmen der Jahrestagung 1997 der Gesellschaft für Informatik* (1997), September

[Boles 1998] BOLES, Dietrich: *Begleitbuch zur Vorlesung Multimedia-Systeme*. Oldenburg : Carl von Ossietzky Universität Oldenburg, Dezember 1998

[Boles u. a. 1998] BOLES, Dietrich ; DAWABI, Peter ; SCHLATTMANN, Marco ; BOLES, Eckhard ; TRUNK, Claudia ; WIGGER, Frank: Objektorientierte Multimedia-Softwareentwicklung: Vom UML-Modell zur Director-Anwendung am Beispiel virtueller naturwissenschaftlich-technischer Labore. In: APPELRATH, Hans-Jürgen (Hrsg.) ; BOLES, Dietrich (Hrsg.) ; MEYER-WEGENER, Klaus (Hrsg.): *Tagungsband zum Workshop Multimedia-Systeme im Rahmen der GI-Jahrestagung 1998*, September 1998, S. 33–51

[Boles und Schlattmann 1998] BOLES, Dietrich ; SCHLATTMANN, Marco: Multimedia-Autorensysteme: Graphisch-interaktive Werkzeuge zur Erstellung multimedialer Anwendungen. In: *LOG IN 18* Heft 1 (1998), S. 10–18

[Boles und Wütherich 1997] BOLES, Dietrich ; WÜTHERICH, Gerd: Transformationelle Multimedia-Softwareentwicklung. In: FUHR, Norbert (Hrsg.) ; DITTRICH, Gisbert (Hrsg.) ; TOCHTERMANN, Klaus (Hrsg.): *Hypertext - Information Retrieval - Multimedia '97: Theorien, Modelle und Implementierungen integrierter elektronischer Informationssysteme; Proceedings HIM '97* Bd. 30. Dortmund : Universitätsverlag Konstanz, 1997, S. 95–108. – Hypertext - Information Retrieval - Multimedia (HIM)

[Brake 2000] BRAKE, Christoph: *Medien in der Wissenschaft*. Bd. 11: *Politikfeld Multimedia : multimediale Lehre im Netz der Restriktionen*. Münster, New York, München, Berlin : Waxmann, 2000

[Brockhaus-Enzyklopädie 1992] BROCKHAUS-ENZYKLOPÄDIE: *Brockhaus Enzyklopädie in vierundzwanzig Bänden*. 1992. – Neunzehnte Auflage

[Brodbeck und Rupietta 1994] BRODBECK, Felix C. ; RUPIETTA, Walter: *Fehlermanagement und Hilfesysteme*. Kap. 5, S. 197–234. In: EBERLEH, Edmund (Hrsg.) ; OBERQUELLE, Horst (Hrsg.) ; OPPERMANN, Reinhard (Hrsg.): *Einführung in die Software-Ergonomie: Gestaltung graphisch-interaktiver Systeme: Prinzipien, Werkzeuge, Lösungen*. Berlin, New York : Walter de Gruyter & Co., 1994

[Bundesdatenschutzgesetz 2001] BUNDESDATENSCHUTZGESETZ: *Bundesdatenschutzgesetz*. Mai 2001. – URL `http://www.bfd.bund.de/`

[Coldewey 2001] COLDEWEY, Jens: Über sieben Brücken musst Du geh'n - Eine Kritik am Software-Engineering. In: *OBJEKTspektrum Nr. 1/2001* (2001)

[Dick 2000] DICK, Egon: *Multimediale Lernprogramme und telematische Lernarrangements : Einführung in die didaktische Gestaltung*. Nürnberg : BW, Bildung und Wissen, Verlag und Software GmbH, 2000 (Multimediales Lernen in der Berufsbildung)

[Dittmar und Eckstein 2001] DITTMAR, Thorsten ; ECKSTEIN, Jutta: Eine Auseinandersetzung mit XP und anderen leichtgewichtigen Methoden. In: *OBJEKTspektrum Nr. 1/2001* (2001)

[D'Souza und Wills 1998] D'SOUZA, Desmond Francis ; WILLS, Alan Cameron: *Objects, components, and frameworks with UML: the Catalysis approach*. Addison Wesley Longman, Inc., Dezember 1998 (The Addison-Wesley object technology series)

[Duden »Informatik« 1993] DUDEN »INFORMATIK« ; CLAUS, Volker (Hrsg.) ; SCHWILL, Andreas (Hrsg.): *Duden »Informatik« : ein Sachlexikon für Studium und Praxis*. 2., vollst. überarb. und erw. Aufl. Mannheim, Leipzig, Wien, Zürich : Dudenverlag - Lektorat des BI-Wiss.-Verl. unter Leitung von Hermann Engesser, 1993

[Eiwan 1999] EIWAN, Barbara: Interaktivität und Lerneffizienz. In: BRAUNGART, Georg (Hrsg.) ; HITZENBERGER, Ludwig (Hrsg.) ; LEHNER, Franz (Hrsg.): *Multimedia : Informationssysteme zwischen Bild und Sprache*. Wiesbaden : Gabler, 1999 (Gabler Edition Wissenschaft : Multimedia und Telekooperation), S. 79–93

[Elfreich 1999] ELFREICH, Sigurd: *Entwicklung eines Java-basierten objektorientierten Frameworks für virtuelle naturwissenschaftliche Experimente im Internet*. Oldenburg : Carl von Ossietzky Universität Oldenburg — Fachbereich Informatik, Dezember 1999. – Diplomarbeit

[Förster und Zwernemann 1993] FÖRSTER, Hans-Peter ; ZWERNEMANN, Martin: *Multimedia - Die Evolution der Sinne!* Neuwied, Kriftel, Berlin : Hermann Lichterhand Verlag GmbH & Co. KG, 1993

[Freibichler 2000] FREIBICHLER, Hans: Protokolle von Lernprozessen. In: LOTTMANN, Alfred (Hrsg.) ; SCHENKEL, Peter (Hrsg.) ; TERGAN, Sigmar-Olaf (Hrsg.): *Qualitätsbeurteilung multimedialer Lern- und Informationssysteme : Evaluationsmethoden auf dem Prüfstand*. Nürnberg : BW, Bildung und Wissen, Verlag und Software GmbH, 2000 (Multimediales Lernen in der Berufsbildung), S. 304–328

[Frühauf u. a. 1991a] FRÜHAUF, Karol ; LUDEWIG, Jochen ; SANDMAYR, Helmut: *Software-Projektmanagement und -Qualitätssicherung*. Stuttgart : B. G. Teubner, 1991 (Leitfäden der angewandten Informatik)

[Frühauf u. a. 1991b] FRÜHAUF, Karol ; LUDEWIG, Jochen ; SANDMAYR, Helmut: *Software-Prüfung: eine Fibel*. Stuttgart : B. G. Teubner, 1991

[Gaines und Shaw 1995] GAINES, Brian R. ; SHAW, Mildred L. G.: *Knowledge Acquisition Tools based on Personal Construct Psychology*. September 1995. – URL `http://ksi.cpsc.ucalgary.ca/articles/KBS/KER/`. – University of Calgary - Knowledge Science Institute

[Gallenberger u. a. 1999] GALLENBERGER, W. ; GRUBER, H. ; HARTEIS, C. ; HEID, H. ; KRAFT, S.: Lehren und Lernen mit neuen Medien. In: BRAUNGART, Georg

(Hrsg.) ; HITZENBERGER, Ludwig (Hrsg.) ; LEHNER, Franz (Hrsg.): *Multimedia : Informationssysteme zwischen Bild und Sprache*. Wiesbaden : Gabler, 1999 (Gabler Edition Wissenschaft : Multimedia und Telekooperation), S. 259–271

[Gamma u. a. 2000] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design Patterns : elements of reusable object-oriented software*. 21th printing. Addison Wesley Longman, Inc., November 2000 (Addison-Wesley professional computing series)

[GenLab - Internetseiten 2001] GENLAB - INTERNETSEITEN: *GenLab - Das virtuelle gentechnische Praktikum*. 2001. – URL `http://www.offis.de/genlab/`

[Groos und Muthmann 1995] GROOS, Stefan ; MUTHMANN, Peter ; ZILAHI-SZABOÓ, Miklós Géza (Hrsg.): *Kleines Lexikon der Informatik und Wirtschaftsinformatik*. München, Wien : R. Oldenbourg Verlag, 1995

[Grupp 1996] GRUPP, Bruno: *Qualifizierung zum Projektleiter: Projektmanagement im Wandel*. 2. Auflage. München : Computerwoche Verlag GmbH, 1996

[Hasler u. a. 2001] HASLER, Anja ; KUCZEWSKI, Ralf ; SCHLATTMANN, Marco: *Multimediaprojekt Gentechnisches Praktikum - Entwicklerleitfaden*. 2001

[Hasler und Schlattmann 2001] HASLER, Anja ; SCHLATTMANN, Marco: *Multimediaprojekt Gentechnisches Praktikum - Endbericht*. 2001

[Heinrich und Schifman 2000] HEINRICH, Günther ; SCHIFMAN, Richard S.: *Multimedia-Projektmanagement: von der Idee zum Produkt*. 2. Auflage. Berlin, Heidelberg, New York, Barcelona, Hongkong, London, Mailand, Paris, Singapur, Tokio : Springer-Verlag, 2000 (X.media.interaktiv)

[Hertlein 1997] HERTLEIN, Margit: *Mind Mapping - Die kreative Arbeitstechnik*. Rowohlt Taschenbuch Verlag GmbH, Juni 1997

[Hesse u. a. 1997] HESSE, Friedrich W. ; GARSOFFKY, Bärbel ; HRON, Aemilian: Interface-Design für computerunterstütztes kooperatives Lernen. In: ISSING, Ludwig J. (Hrsg.) ; KLIMSA, Paul (Hrsg.): *Information und Lernen mit Multimedia*. 2., überarbeitete Auflage. Weinheim : Psychologie Verlags Union, Januar 1997, S. 253–267

[Heuer und Saake 2000] HEUER, Andreas ; SAAKE, Gunter: *Datenbanken: Konzepte und Sprachen*. 2. Auflage. Bonn : MITP-Verlag, 2000

[Heuten 2001] HEUTEN, Wilko: *3D Visualisierungskonzepte für virtuelle Labore*. Oldenburg : Carl von Ossietzky Universität Oldenburg — Fachbereich Informatik, Februar 2001. – Diplomarbeit

[Hruschka u. a. 2001] HRUSCHKA, Peter ; JOSUTTIS, Nicolai ; KOCHER, Hartmut ; KRASEMANN, Hartmut ; OESTEREICH, Bernd ; REINHOLD, Markus ; OESTEREICH, Bernd (Hrsg.): *Erfolgreich mit Objektorientierung - Vorgehensmodelle und Managementpraktiken*

für die objektorientierte Softwareentwicklung. 2., aktualisierte und ergänzte Auflage. München : R. Oldenbourg Verlag, 2001

[IABG 2001] IABG: *Das V-Modell*. 2001. – URL http://www.v-modell.iabg.de/

[IEEE Learning Technology Standards Committee 2000] IEEE LEARNING TECHNOLOGY STANDARDS COMMITTEE: *IEEE P1484.3/D2 Draft Standard for Information Technology — Learning Technology — Glossary*. 2000. – URL http://ltsc.ieee.org/

[Internet2 1997–2000] INTERNET2: *The Virtual Laboratory: An Application Environment for Computational Science and Engineering*. 1997-2000. – URL http://www.internet2.edu/html/virtual_laboratory.html

[Issing 1997] ISSING, Ludwig J.: Instruktionsdesign für Multimedia. In: ISSING, Ludwig J. (Hrsg.) ; KLIMSA, Paul (Hrsg.): *Information und Lernen mit Multimedia*. 2., überarbeitete Auflage. Weinheim : Psychologie Verlags Union, Januar 1997, S. 241–251

[Jacob und Kronberg 1999] JACOB, Ch. ; KRONBERG, Manfred: Virtuelles Praktikum Leistungselektronik. In: *3. Workshop Multimedia für Bildung und Wirtschaft* (1999), September

[Jacobson u. a. 1999] JACOBSON, Ivar ; BOOCH, Grady ; RUMBAUGH, James: *The unified software development process*. Addison-Wesley, 1999 (Addison-Wesley Object Technology Series)

[Jarz 1997] JARZ, Ewald M.: *Entwicklung multimedialer Systeme : Planung von Lern- und Masseninformationssystemen*. Wiesbaden : Deutscher Universitäts-Verlag, Gabler Verlag, 1997 (Gabler Edition Wissenschaft)

[Kerres 1998] KERRES, Michael: *Multimediale und telemediale Lernumgebungen : Konzeption und Entwicklung*. Wien, München : R. Oldenbourg Verlag, 1998

[Klimsa 1997] KLIMSA, Paul: Multimedia aus psychologischer und didaktischer Sicht. In: ISSING, Ludwig J. (Hrsg.) ; KLIMSA, Paul (Hrsg.): *Information und Lernen mit Multimedia*. 2., überarbeitete Auflage. Weinheim : Psychologie Verlags Union, Januar 1997, S. 7–23

[Knoblich 1998] KNOBLICH, Carsten: *Ein Client/Server-System zur Organisation von Strategiespiel-Turnieren im Internet*. Oldenburg : Carl von Ossietzky Universität Oldenburg — Fachbereich Informatik, Mai 1998. – Diplomarbeit

[Kronberg 1998] KRONBERG, Manfred: Virtuelles Praktikum Leistungselektronik. In: *2. Workshop Multimedia für Bildung und Wirtschaft* (1998), September

[Kruchten 2000] KRUCHTEN, Philippe: *The Rational Unified Process : an introduction*. 2nd printing. Addison Wesley, 2000 (Addison-Wesley Object Technology Series)

[Kunz und Schott 1987] KUNZ, Gunnar C. ; SCHOTT, Franz: *Intelligente Tutorielle Systeme - Neue Ansätze der computerunterstützten Steuerung von Lehr-Lern-Prozessen*. Göttingen, Toronto, Zürich : Verlag für Psychologie - C.J. Hogrefe, 1987

[Langenscheidts Fremdwörterbuch 2001] LANGENSCHEIDTS FREMDWÖRTERBUCH: *Langenscheidts Fremdwörterbuch online*. 2001. – URL `http://www.langenscheidt.aol.de/`

[Larman u. a. 2001] LARMAN, Craig ; KRUCHTEN, Philippe ; BITTNER, Kurt: Wie Sie ein Projekt mit dem Rational Unified Process garantiert in den Sand setzen können. In: *OBJEKTspektrum Nr. 4/2001* (2001)

[Leutner 1997] LEUTNER, Detlev: Adaptivität und Adaptierbarkeit multimedialer Lehr- und Informationssysteme. In: ISSING, Ludwig J. (Hrsg.) ; KLIMSA, Paul (Hrsg.): *Information und Lernen mit Multimedia*. 2., überarbeitete Auflage. Weinheim : Psychologie Verlags Union, Januar 1997, S. 138–139

[Liening 1992] LIENING, Andreas ; KRAFFT, Dietmar (Hrsg.): *Ökonomische Bildung*. Bd. 1: *Intelligente Tutorielle Systeme : (ITS) ; eine kritische Analyse unter besonderer Berücksichtigung ihrer Einsatzmöglichkeiten in der Wirtschaftsdidaktik*. Hamburg : Lit Verlag, 1992. – Dissertation

[Liggesmeyer 1990] LIGGESMEYER, Peter: *Modultest und Modulverifikation: state of the art*. Mannheim, Wien, Zürich : BI-Wissenschaftsverlag, 1990 (Angewandte Informatik Band 4)

[Liggesmeyer 1993] LIGGESMEYER, Peter: *Wissensbasierte Qualitätsassistenz zur Konstruktion von Prüfstrategien für Software-Komponenten*. Mannheim, Leipzig, Wien, Zürich : BI-Wissenschaftsverlag, 1993 (Forschung in der Softwaretechnik Band 1)

[Lottmann u. a. 2000] LOTTMANN, Alfred (Hrsg.) ; SCHENKEL, Peter (Hrsg.) ; TERGAN, Sigmar-Olaf (Hrsg.): *Qualitätsbeurteilung multimedialer Lern- und Informationssysteme : Evaluationsmethoden auf dem Prüfstand*. Nürnberg : BW, Bildung und Wissen, Verlag und Software GmbH, 2000 (Multimediales Lernen in der Berufsbildung)

[Lusti 1992] LUSTI, Markus ; ENDRES, Albert (Hrsg.): *Handbuch der Informatik*. Bd. 15.4: *Intelligente tutorielle Systeme : Einführung in wissensbasierte Lernsysteme*. München, Wien : Oldenbourg, 1992

[Merx 1999] MERX, Oliver (Hrsg.): *Qualitätssicherung bei Multimedia-Projekten*. Berlin, Heidelberg, New York, Barcelona, Hongkong, London, Mailand, Paris, Singapur, Tokio : Springer-Verlag, 1999 (X.media.interaktiv)

[Nagl u. a. 1999] NAGL, M. ; BEHLE, A. ; WESTFECHTEL, B. ; BALZERT, H. ; WEIDAUER, C. ; SIX, H.-W. ; PAUEN, P. ; VOSS, J. ; SCHÄFER, W. ; WADSACK, J. ; KELTER, U.: *Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme*. September 1999. – Studie

[Nowaczyk 2000] NOWACZYK, Olaf: Interaktive Lernmodule für die Technische Mechanik nach dem Explorationen-Konzept. In: *4. Workshop Multimedia für Bildung und Wirtschaft* (2000), September

[Object Management Group 1997] OBJECT MANAGEMENT GROUP: *UML Summary version 1.1*. 1997

[Object Management Group 1997–2001] OBJECT MANAGEMENT GROUP: *UML Resource Page*. 1997-2001. – URL http://www.omg.org/uml/

[Oestereich 1999] OESTEREICH, Bernd: Die Macht des Rhythmus: das OEP-Timepacing-Verfahren. In: *OBJEKTspektrum Nr. 2/2000* (1999)

[oose.de GmbH 1999] OOSE.DE GMBH: *Software entwickeln mit System: Object Engineering Process*. 1999. – URL http://www.oose.de/oep/

[Oppermann und Reiterer 1994] OPPERMANN, Reinhard ; REITERER, Harald: *Software-ergonomische Evaluation*. Kap. 9, S. 335–371. In: EBERLEH, Edmund (Hrsg.) ; OBERQUELLE, Horst (Hrsg.) ; OPPERMANN, Reinhard (Hrsg.): *Einführung in die Software-Ergonomie: Gestaltung graphisch-interaktiver Systeme: Prinzipien, Werkzeuge, Lösungen*. Berlin, New York : Walter de Gruyter & Co., 1994

[Pagel und Six 1994] PAGEL, Bernd-Uwe ; SIX, Hans-Werner: *Software Engineering: Die Phasen der Softwareentwicklung*. Addison-Wesley, 1994

[Pasch 1994] PASCH, Jürgen: *Kooperative Software-Entwicklung: mehr Qualität durch das dialogische Prinzip bei der Projektarbeit*. Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong, Barcelona, Budapest : Springer-Verlag, 1994

[Preim 1999] PREIM, Bernhard: *Entwicklung interaktiver Systeme: Grundlagen, Beispiele und innovative Anwendungsfelder*. Berlin, Heidelberg, New York, Barcelona, Hongkong, London, Mailand, Paris, Singapur, Tokio : Springer-Verlag, 1999

[Projektgruppe Elvis - Internetseiten 1998] PROJEKTGRUPPE ELVIS - INTERNETSEITEN: *Projekt Elvis*. 1998. – URL http://elvis.offis.uni-oldenburg.de/

[Psaralidis und Zimmer 2000] PSARALIDIS, Elena ; ZIMMER, Gerhard: Der Lernerfolg bestimmt die Qualität einer Lernsoftware! Evaluation von Lernerfolg als logische Rekonstruktion von Handlungen. In: LOTTMANN, Alfred (Hrsg.) ; SCHENKEL, Peter (Hrsg.) ; TERGAN, Sigmar-Olaf (Hrsg.): *Qualitätsbeurteilung multimedialer Lern- und Informationssysteme : Evaluationsmethoden auf dem Prüfstand*. Nürnberg : BW, Bildung und Wissen, Verlag und Software GmbH, 2000 (Multimediales Lernen in der Berufsbildung), S. 262–303

[Reimer 1991] REIMER, Ulrich: *Einführung in die Wissensrepräsentation : netzartige und schema-basierte Repräsentationsformate*. Stuttgart : B.G. Teubner, 1991

[Reißing 2000] REISSING, Ralf: Extremes Programmieren. In: *Informatik Spektrum* Band 23 (2000), April, Nr. 2

[Ritter 2000] RITTER, Jörg: *Prozessorientierte Konfiguration komponentenbasierter Anwendungssysteme*. Oldenburg : Carl von Ossietzky Universität Oldenburg — Fachbereich Informatik, Mai 2000. – Dissertation zur Erlangung des Grades des Doktor der Naturwissenschaften

[Rupietta 1987] RUPIETTA, Walter: *Benutzerdokumentation für Softwareprodukte*. Mannheim, Wien, Zürich : BI-Wissenschaftsverlag, 1987 (Angewandte Informatik Band 3)

[Russell und Norvig 1995] RUSSELL, Stuart Jonathan ; NORVIG, Peter: *Artificial Intelligence: a modern approach*. Prentice-Hall, Inc., 1995

[Sawhney 1995] SAWHNEY, Monica: *Entwicklung eines Vorgehensmodells für die Multimedia-Anwendungsentwicklung am Beispiel eines Informations- und Orientierungssystems für eine Universität*. Osnabrück : Universität Osnabrück — Fachbereich Wirtschaftswissenschaften, Juni 1995. – Freie wissenschaftliche Arbeit zur Erlangung des Grades einer Diplom-Kauffrau

[Schanda 1995] SCHANDA, Franz: *Computer-Lernprogramme: wie damit gelernt wird ; wie sie entwickelt werden ; was sie im Unternehmen leisten*. Weinheim, Basel : Beltz Verlag, 1995

[Schmidt 1989] SCHMIDT, Frank: *Expertensystem-Erstellung unter der Berücksichtigung von Erkenntnissen der kognitiven Psychologie und verschiedener Knowledge engineering Methoden zur Extraktion von Expertenwissen*. Kassel : Universität Kassel, 1989. – Dissertation

[Schröder 1996] SCHRÖDER, O.: Intelligente tutorielle Systeme / Intelligente computerunterstützte Instruktion. In: BECKER, Barbara (Hrsg.) ; FREKSA, Christian (Hrsg.) ; HAHN, Udo (Hrsg.) ; OPWIS, Klaus (Hrsg.) ; PALM, Günther (Hrsg.) ; STRUBE, Gerhard (Hrsg.): *Wörterbuch der Kognitionswissenschaft*. Stuttgart : Klett-Cotta, 1996, S. 289–291

[Schulmeister 1997] SCHULMEISTER, Rolf: *Grundlagen hypermediale Lernsysteme: [Theorie - Didaktik - Design]*. 2., aktualisierte Auflage. München, Wien : R. Oldenbourg Verlag, 1997

[Schult 1994] SCHULT, Thomas J.: Bücher zerstören die Erziehung: Roger Schank im Gespräch mit Thomas J. Schult. In: *c't* (1994), Nr. 1/1994, S. 56–58

[Sommerville 1996] SOMMERVILLE, Ian: *Software Engineering*. 5th edition, reprinted. Addison-Wesley, 1996

[Steinhausen 1994] STEINHAUSEN, Detlef: *Simulationstechniken*. München, Wien : Oldenbourg, 1994

[Steinmetz 1999] STEINMETZ, Ralf: *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Berlin, Heidelberg, New York, Barcelona, Hongkong, London, Mailand, Paris, Singapur, Tokio : Springer-Verlag, 1999

[Strittmatter und Mauel 1997] STRITTMATTER, Peter ; MAUEL, Dirk: Einzelmedium, Medienverbund und Multimedia. In: ISSING, Ludwig J. (Hrsg.) ; KLIMSA, Paul (Hrsg.): *Information und Lernen mit Multimedia*. 2., überarbeitete Auflage. Weinheim : Psychologie Verlags Union, Januar 1997, S. 47–61

[Tanenbaum 1996] TANENBAUM, Andrew S.: *Computer Networks*. Third Edition. Prentice-Hall, Inc., 1996

[Thissen 2000] THISSEN, Frank: *Screen-Design-Handbuch: Effektiv informieren und kommunizieren mit Multimedia*. Berlin, Heidelberg, New York, Barcelona, Hongkong, London, Mailand, Paris, Singapur, Tokio : Springer-Verlag, 2000

[Virtual Physiology - Internetseiten 2001] VIRTUAL PHYSIOLOGY - INTERNETSEITEN: *Georg Thieme Verlag: Die Reihe Virtual Physiology*. 2001. – URL `http://www.thieme.de/elm/sim/index.html`

[ViSeL - Internetseiten 2001] VISEL - INTERNETSEITEN: *ViSeL - Virtual Sequencing Laboratory*. 2001. – URL `http://www.TechFak.Uni-Bielefeld.DE/ags/pi/ViSeL/`

[Witschital 1990] WITSCHITAL, Peter: *Intelligente Tutorielle Systeme in der Programmierausbildung*. Hildesheim : Universität Hildesheim - Institut für Betriebssysteme & Rechnerverbund, Juli 1990. – Dissertation

[Yass 2000] YASS, Mohammed: *Entwicklung multimedialer Anwendungen : eine systematische Einführung*. Heidelberg : dpunkt.verlag GmbH, 2000

[Zehnder 1991] ZEHNDER, Carl August: *Informatik-Projektentwicklung*. 2., überarbeitete und erweiterte Auflage. Stuttgart : B. G. Teubner, 1991 (Leitfäden der angewandten Informatik)

[Ziegler 1993] ZIEGLER, Jürgen: *Entwurf graphischer Benutzungsschnittstellen*. Kap. 9. In: ILG, Rolf (Hrsg.) ; ZIEGLER, Jürgen (Hrsg.): *Benutzergerechte Software-Gestaltung : Standards, Methoden und Werkzeuge*. München, Wien : R. Oldenbourg Verlag, 1993

Index

O

P

Q

R

W

X

Z